

AD-A133 222

VALIDATION OF DIGITAL SYSTEMS IN AVIONICS AND FLIGHT
CONTROL APPLICATIONS. (U) BATTELLE COLUMBUS LABS OH
E F HITT ET AL. JUL 83 DOT/FAA/CT-82/115

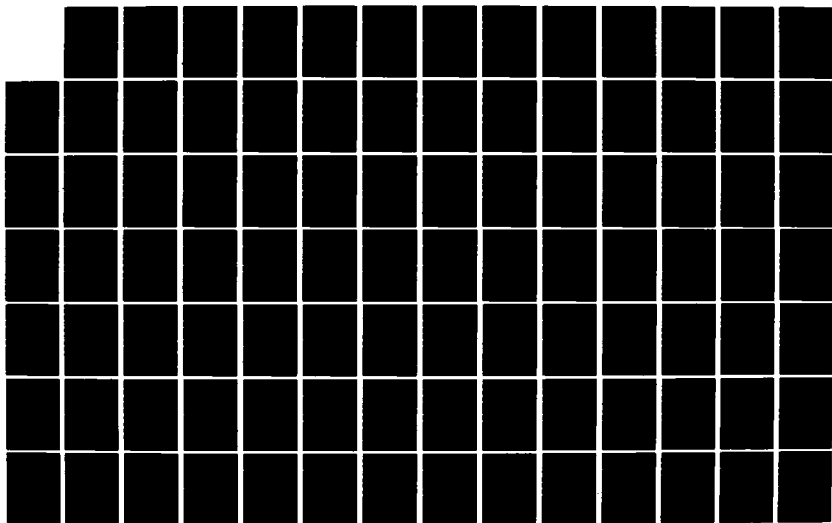
1/5

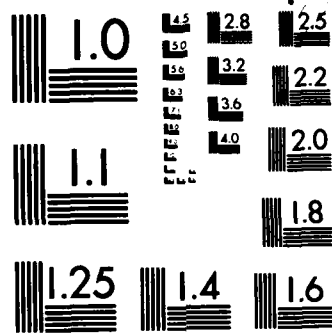
UNCLASSIFIED

DTFA03-81-C-00059

F/G 1/3

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A133 222

15

DOT/FAA/CT-82/115

Handbook – Volume I Validation of Digital Systems in Avionics and Flight Control Applications

Ellis F. Hitt
Jeff Webb
Charles Lucius
Michael S. Bridgman
Battelle Columbus Laboratories
Columbus, Ohio 43201

Donald Eldredge
FAA Technical Center
Atlantic City Airport, New Jersey 08405

July 1983

This document is available to the U.S. public
through the National Technical Information
Service, Springfield, Virginia 22161.

DTIC FILE COPY



US Department of Transportation
Federal Aviation Administration
Technical Center
Atlantic City Airport, N.J. 08405

DTIC

OCT 04 1983

E

83 09 26 051

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturer's names appear herein solely because they are considered essential to the object of this report.

Technical Report Documentation Page

1. Report No. DOT/FAA/CT-82/115		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle HANDBOOK--VOLUME I, VALIDATION OF DIGITAL SYSTEMS IN AVIONICS AND FLIGHT CONTROL APPLICATIONS				5. Report Date	
				6. Performing Organization Code FAA Technical Center	
7. Author(s) E. Hitt, J. Webb, C. Luicus, M. Bridgman, D. Eldredge				8. Performing Organization Report No. DOT/FAA/CT-82/115	
9. Performing Organization Name and Address Battelle-Columbus Laboratories 505 King Avenue Columbus, Ohio 43201				10. Work Unit No. (TRAIS)	
				11. Contract or Grant No. DTFA03-81-C-00059	
12. Sponsoring Agency Name and Address U.S. Department of Transportation Federal Aviation Administration Technical Center Atlantic City Airport, NJ 08405				13. Type of Report and Period Covered Draft Report June 1981 - July 1982	
				14. Sponsoring Agency Code ACT-340	
15. Supplementary Notes					
16. Abstract The purpose of this handbook is to identify techniques, methodologies, tools, and procedures in a systems context that may be applicable to aspects of the validation and certification of digital systems at specific times in the development and certification portion of the system life cycle. The application of these techniques in the development of discrete units and/or systems will result in a completion of a product or system which is verifiable and can be validated in the context of the existing regulations/orders for the government regulatory agencies. The handbook uses a systems engineering approach to the integration and testing of software and hardware during the design, development, and implementation phases. The handbook also recognizes and provides for the evaluation of the pilot's workload and utilization of the new control/display technologies, especially when crew recognition and intervention may be necessary to cope with/recover from the effects of faults or failures in the digital systems. In summary, the handbook: (1) Identifies and presents the issues related to the design, development, and implementation of software based digital systems; (2) identifies specific approaches applicable to all aspects of the verification and validation procedures, at specific times in the development and certification portion of the system life cycle, (3) provides the government regulatory agencies (especially the Federal Aviation Administration (FAA) as well as the industry with a set of tools/procedures, in a systems engineering context which may be of value in the validation/certification process.					
17. Key Words Validation/Verification, Airborne Systems Digital Systems, Software Reliability, Software Testing, System Evaluation, Digital Flight Control/Avionics, Safety Assessment, Configuration Management,				18. Distribution Statement	
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of Pages	
				22. Price	

TABLE OF CONTENTS

SECTION 1

1. INTRODUCTION

- 1.1 Purpose
- 1.2 Distribution
- 1.3 Background
- 1.4 Handbook Scope
- 1.5 Handbook Organization
- 1.6 References

SECTION 2

2. Applicable Documents

SECTION 3

3. SYSTEM LIFE CYCLE (OVERVIEW)

- 3.1 Operational Flight Program (OFP) Development
 - 3.1.1 Conceptual
 - 3.1.2 Requirements Definition
 - 3.1.3 Design and Specification
 - 3.1.4 Coding
 - 3.1.5 Testing
 - 3.1.6 Verification, Validation, Certification, and Qualification
- 3.2 Function Criticality
 - 3.2.1 Criticality Assessment
- 3.3 Potential Problem Areas
- 3.4 References

Accession For	
NTIS GR&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution	
Availability Codes	
Dist	
A	

Page

1-1

1-1

1-1

1-1

1-2

1-7

1-8

1-10

2-1

2-1

3-1

3-1

3-3

3-3

3-3

3-6

3-8

3-8

3-13

3-17

3-17

3-17

3-20

LIST OF ILLUSTRATIONS

Figure

Page

3-1 System Life Cycle

3-1

3-2 Software Activities/Products Relation to System Life Cycle Phases

3-2

3-3	Avionics Software Engineering Process	3-4
3-4	Test Management Milestones	3-7
3-5	Software Activities/Products Relation to Verification/ Validation Activities	3-15
3-6	Verification and Validation Activities by Organization Over Software Life Cycle (Ref. 22)	3-16

LIST OF TABLES

Table		Page
3-1	Function Criticality Categories	3-18
3-2	Minimum Verification and Validation Activities Grouped by Criticality (Ref. 24)	3-19
SECTION 4		4-1
4. MISSION FACTORS		4-1
4.1 Mission Environment		4-1
4.1.1 Airline Operations		4-1
4.1.2 Airport Configuration		4-4
4.1.3 Air Traffic Mix/Density at Specific Airports as a Function of Time-of-Day		4-4
4.1.4 Air Traffic Control/Navigation Aids		4-4
4.1.5 Operating Time Considerations in Validation		4-5
4.2 Atmospheric Environment		4-6
4.2.1 Lightning		4-7
4.3 Electromagnetic Interference		4-8
4.4 Safety Requirements		4-8
4.5 Constraints		4-15
4.5.1 Aircraft		4-15
4.5.2 Airport		4-15
4.5.3 Approach/Landing Performance Limits		4-15
4.5.4 Airline Imposed Constraints		4-17
4.6 Advanced Digital Integrated Flight Control and Avionics System Functions		4-17
4.6.1 Flight Control Background		4-17
4.6.2 Command Generation (Guidance) Background		4-17

4.6.3	State Estimation (Navigation) Background	4-18
4.6.4	Aircraft System Functions	4-19
4.7	References	4-22

LIST OF ILLUSTRATIONS

Figure		Page
4-1	Flux Linkages Versus Conductor Position (Reference 12)	4-9
4-2	Conductor Routing (Reference 12)	4-10
4-3	Frequency Considerations	4-10
4-4	Relationship Between Probability and Severity of Effects (Reference 29)	4-13
4-5	Block Diagram of Navigation, Guidance, and Control Outer Loop	4-20

LIST OF TABLES

Table		Page
4-1	Equipment Required for IFR Dispatch (FAR Parts 25.1303 and 121.303)	4-2
4-2	Airborne Equipment Requirements Categories (Categories I and II ⁺)	4-3
4-3	Primary Radio Navigation Systems	4-5
4-4	Example of Digital Flight Control and Avionics System Operating Profile	4-6
4-5	Protection from Indirect Electromagnetic Effects	4-8
4-6	EMI Sources and Susceptibility of a Hypothetical Digital Avionics Subsystem	4-11
4-7	How to Improve EMC of the Subsystem	4-12
4-8	Quantitative Safety Requirements	4-12
SECTION 5		5-1
5.	SYSTEM ARCHITECTURES	5-1
5.1	Fault Tolerant Digital Integrated Flight Control and Avionics	5-1

5.1.1	Redundancy Techniques	5-4
5.1.2	Motivation for Fault Tolerant Integrated Control	5-5
5.2	Advanced Integrated Digital Flight Control and Avionics Systems	5-6
5.2.1	Digital Data Buses	5-7
5.3	Software	5-11
5.3.1	Coordinate Systems	5-11
5.3.2	Integrated Control Core Software Modules	5-14
5.4	System Monitoring and Error Management	5-14
5.4.1	Processor Failure Detection	5-14
5.4.2	Data Transmission Error Detection	5-16
5.4.3	Data Validity	5-17
5.5	Systems Configuration	5-18
5.5.1	Simplex Configuration	5-19
5.5.2	Single-String Configurations	5-19
5.5.3	Dual System Configuration	5-20
5.5.4	Dual-Dual System Configuration	5-23
5.5.5	Triplex	5-25
5.6	ARINC Subsystems	5-27
5.6.1	Area Navigation System	5-27
5.6.2	Flight Control Computer System	5-31
5.6.3	Flight Management Computer System	5-39
5.6.4	Thrust Control Computer System	5-39
5.6.5	Inertial Reference System (IRS)	5-45
5.6.6	Attitude and Heading Reference System	5-45
5.6.7	Air Data System	5-51
5.6.8	Radio Altimeter	5-51
5.6.9	Airborne Weather Radar	5-51
5.6.10	Airborne Distance Measuring Equipment	5-55
5.6.11	ILS Receiver	5-55
5.6.12	Airborne VOR Receiver	5-56
5.6.13	Airborne ADF System	5-56
5.6.14	Mark 2 Omega Navigation System	5-56
5.6.15	Airborne VHF Communication Transceiver	5-56
5.6.16	Flight Data Acquisition and Recording System	5-57
5.6.17	Mark 3 Air Traffic Control Transponder (ATCRBS/DABS)	5-57
5.6.18	Digital Frequency/Function Selection for Airborne Electronic Equipment	5-57
5.6.19	Ground Proximity Warning System	5-57
5.6.20	Electronic Flight Instruments (EFI)	5-59
5.6.21	Flight Warning Computer System	5-60
5.6.22	Airborne MLS Receiver	5-60
5.6.23	Analog and Discrete Data Converter System	5-60

5.6.24	Airborne Separation Assurance System	5-60
5.6.25	Electric Chronometer	5-61
5.6.26	Digital Engine Controller	5-61
5.6.27	Electric Power Generation System	5-61
5.6.28	Synopsis of Digital Interfaces of ARINC Characteristic Subsystems	5-62

5.7	References	5-67
-----	------------	------

LIST OF ILLUSTRATIONS

Figure		Page
5-1	Fault Tolerance Definitional Relationships	5-2
5-2	Fault Types	5-2
5-3	ARINC 429 General Word Format	5-7
5-4	MIL-STD-1553B Word Format	5-10
5-5	MIL-STD-1553B Information Transfer Formats	5-10
5-6	Inertial Coordinate System	5-12
5-7	Earth Fixed Coordinate System	5-12
5-8	Body Axes Coordinate System	5-13
5-9	Locally Level Coordinate System	5-13
5-10	Horizontal Plane Coordinate System	5-15
5-11	Flight Management Computer System Configuration 2: Single System/Dual CDU Installation	5-20
5-12	Flight Management Computer System Configuration 3: Dual System Installation	5-21
5-13	Dual Digital Flight Control System With Inter-Unit Switching (Ref. 39)	5-22
5-14	Typical Dual-Dual AFS Configuration	5-24
5-15	Typical Triplex DAFS Architecture	5-26
5-16	ARINC 701 Interfaces	5-32
5-17	Flight Management Computer System Interface With Flight Control System and Other Subsystems	5-40
5-18	ARINC 703 Thrust Control Computer Interface	5-44

5-19	704 Block Diagram	5-49
------	-------------------	------

LIST OF TABLES

Table		Page
5-1	Dits Buses	5-8
5-2	ARINC Characteristic 581 Functions	5-28
5-3	ARINC 581 System Inputs - Range and Resolution	5-28
5-4	ARINC Characteristic 581 System Operation With Failed Sensors	5-29
5-5	ARINC Characteristic 583 Basic Functions	5-30
5-6	ARINC 583 System Inputs Range and Resolution	5-30
5-7	Example of ARINC Characteristic 583 System Operation With Failed Sensors	5-31
5-8	ARNIC 701 Flight Control Computer (FCC) System Controller Functions	5-36
5-9	FCCS Data Words	5-37
5-10	FCCS Discrete Input and Output Word Formats (Discrete Word #1)	5-37
5-11	FCCS Discrete Word #2	5-38
5-12	ARINC 702 Flight Management System Functions	5-41
5-13	Thrust Control Computer System Components and Functions	5-43
5-14	IRS Digital Summary - Inputs/Outputs (Reference 51)	5-45
5-15	Digital Control Word Format IRS Discrete Word Format	5-47
5-16	AHRS Digital Output Summary	5-50
5-17	Air Data System Digital Data Output Standards (Reference 53)	5-52
5-18	ADS Discrete Word #1 Format (Reference 53)	5-53
5-19	ADS Discrete Word #2 Format (Reference 53)	5-54
5-20	709 DME Mode Selection Matrix	5-55
5-21	Discrete Work Format (Reference 66)	5-58
5-22	ARINC 725-1 Signal Generator Digital Inputs (Reference 67)	5-59

5-23	ARINC 726-1 Flight Warning Computer System Digital Inputs (Reference 68)	5-60
5-24	ARINC 730-3 Airborne Separation Assurance System Digital Inputs (Reference 71)	5-61
5-25	Synopsis of Digital Interfaces of ARINC Characteristic Subsystems	5-63
SECTION 6		6-1
6.	CREW WORKLOAD EVALUATION	6-1
6.1	Human Factors Engineering Goals	6-1
6.1.1	Safety and Crew Acceptability	6-1
6.1.2	Current Revolutionary Changes in Crew Interface	6-2
6.1.3	Greater Future Design Commonality	6-3
6.2	Regulatory Requirements	6-3
6.2.1	Minimum Crew Determination	6-3
6.2.2	Overall Flight-Deck Evaluation	6-5
6.3	Current Workload Measurement Techniques	6-7
6.3.1	Subjective Measures	6-7
6.3.2	Performance Measures	6-7
6.3.3	Physiological and Biological Measures	6-8
6.4	Automation and System Effectiveness	6-9
6.4.1	Impact on the Pilot	6-10
6.4.2	System Effectiveness	6-12
6.4.3	Changed Pilot Roles	6-14
6.4.4	Human Needs and Satisfaction	6-16
6.5	Future Workload Measurement Problems	6-17
6.5.1	Evaluation of Stress	6-17
6.5.2	Sources of Pilot Error	6-19
6.5.3	Definitions of Good Design	6-20
6.5.4	Relation of Workload to Safety	6-21
6.6	Recent Certification Programs Overview	6-23
6.6.1	Systems Analysis	6-23
6.6.2	Synthetic Performance Measures	6-25
6.6.3	Flight Test	6-29
6.6.4	Remaining Problems	6-35
6.6.5	Need to Expand the Variety of Procedure	6-35
6.6.6	Classification of Workload Assessment Methods	6-38
6.7	Recommended Application	6-42

6.7.1	Specific Procedures	6-42
6.7.2	Combined Methods	6-45
6.8	References	6-47

APPENDIX

LIST OF ILLUSTRATIONS

Figure		Page
6-1	Representative Elements in Goal-Effective System Operation	6-13
6-2	The Pilot-Aircraft-Environment Information Flow	6-18
6-3	Relation of High and Low Pilot Workload and Safety	6-22
6-4	DC-9-80 Comparison of Equipment Interface Workload with DC-9-50 When Autothrottles are Inoperative	6-28
6-5	Varieties of Digit Entry Devices	6-36
6-6	Inventory of Available Workload Methods and Issue/Application Situations	6-43
6-7	Population of Methods of Evaluating Pilot Workload	6-44

SECTION 7	7-1
-----------	-----

7. ISSUES	7-1
7.1 Definition of Verification and Validation	7-1
7.2 Means of Showing Compliance With Air Systems and Worthiness Requirements	7-2
7.3 Function Criticality	7-2
7.3.1 Function Performance Requirements	7-2
7.4 Maintenance of Software	7-4
7.5 Regulatory Review Period	7-4
7.6 Documentation	7-5
7.6.1 Software Documentation Standards	7-5
7.7 Software Configuration Management Control	7-5
7.7.1 Support Software Configuration Control	7-5
7.7.2 Configuration Management/Control of the Operational Flight Program	7-5

7.8	Higher Order Language	7-8
7.9	Redundant Systems Software	7-8
7.10	Test Boundaires	7-8
7.11	Role of Models in Validation	7-8
7.11.1	Analytic Reliability Model Issues	7-8
7.11.2	Scope of Analytic Models	7-9
7.11.3	Verification and Validation of Models	7-9
7.11.4	Interpretation of Model Results	7-10
7.11.5	Numerical Accuracy	7-10
7.11.6	Data Validity	7-11
7-12	Role of Simulation and Testing in Validation	7-11
7-13	References	7-16

LIST OF ILLUSTRATIONS

Figure		Page
7-1	Example of Documentation Tree	7-6
7-2	Simulator-Based System Validation Process (Reference 44)	7-14

LIST OF TABLES

Table		Page
7-1	Quantitative Safety Requirements	7-3
7-2	Digital Flight Control-Basic Function Reliability (Reference 16)	7-4
7-3	System Documentation	7-7
7-4	Example of Iron Bird Simulation Use (Reference 43)	7-12
7-5	Characteristics to be Addressed in the Validation Process (Reference 44)	7-13
7-6	Areas of FAR 25 that are Prime Candidates for Increased Usage of Simulation for Certification (Reference 45)	7-15

SECTION 8	8-1
8. CONCEPTS/METHODOLOGIES	8-1

8.1	Overview	8-1
8.2	Analogy Methods	8-1
8.2.1	Good Engineering Judgment	8-1
8.3	Analytic Models and Methods	8-1
8.3.1	Terminology	8-4
8.3.2	Scope of Analytic Models and Methods	8-4
8.3.3	Criteria for Consideration of Analytic Models and Methods	8-4
8.3.4	Classes of Models and Methods	8-6
8.3.5	Role of Models and Methods in System Development	8-8
8.4	Software Verification/Testing	8-10
8.4.1	Life Cycle/Verification Activities	8-11
8.4.2	Manual Verification Methodologies	8-14
8.4.3	Computer-Aided Methodologies	8-17
8.4.4	Software Execution	8-24
8.4.5	Software Verification/Testing Conclusions	8-37
8.5	Hardware Verification/Testing	8-39
8.5.1	Bus Testing	8-39
8.5.2	ARINC 429-6	8-40
8.5.3	MIL-STD-1553/B	8-45
8.6	System Integration/Verification/Validation	8-53
8.6.1	Ground Simulation	8-53
8.6.2	Hot Bench	8-54
8.6.3	Iron Bird	8-55
8.6.4	Airborne Simulation	8-56
8.6.5	Flight Test	8-56
8.7	References	8-57

LIST OF ILLUSTRATIONS

Figure		Page
8-1	Methodologies Application in Validation of Digital Systems (2 Sheets)	8-2
8-2	Pitch Servo Fault Tree (Reference 3)	8-7
8-3	Typical Elements of an Automatic Test Analyzer (Reference 75)	8-23
8-4	429 Input/Output Circuit Standards	8-41

8-5	429 Output Signal Timing Tolerances	8-42
8-6	ARINC 429 Bus Voltages	8-43
8-7	Basic ARINC 429 Word	8-43
8-8	Stubs in a 429 Bus System	8-44
8-9	Data Encoding Manchester II	8-48
8-10	MIL-STD-1553B Word Format	8-49
8-11	Intermessage Gap and Response Time	8-50
8-12	Information Transfer Formats	8-50
8-13	Hot Bench Facility	8-54
8-14	Iron Bird Overview	8-55

LIST OF TABLES

Table		Page
8-1	Generation of Cut Sets (Reference 3)	8-9
8-2	Criteria for Evaluating Verification Methods	8-11
8-3	Life Cycle Verification Activities	8-12
8-4	Generic Heading for Static Tools and Techniques	8-19
8-5	Some Available Static and/or Dynamic Analyzers (2 Sheets)	8-20
8-6	Generic Headings for Dynamic Tools and Techniques	8-22
8-7	Software Testing Philosophies	8-25
8-8	Module Integration Testing Strategies	8-30
8-9	Types of System Tests	8-36
8-10	Examples of Errors Detected by Each Verification Activity (Reference 86)	8-38
8-11	Parameter Variation Versus Impact on Word Error Rate Experienced on Space Shuttle Program	8-40
8-12	Characteristics of ARINC 429-6	8-42
8-13	Comparison of Data Bus Characteristics (2 Sheets)	8-46
8-14	Comparison of Terminal Characteristics (2 Sheets)	8-51

SECTION 9	9-1
9. CURRENT VALIDATION PROCEDURES	9-1
9.1 Design Validation	9-1
9.1.1 Theoretical Reliability Analysis	9-1
9.1.2 Electronic Design Verification	9-2
9.1.3 Failure Modes and Effects Analysis	9-2
9.1.4 Sneak Analysis	9-4
9.2 Hardware Testing	
9.2.1 Development Tests	9-6
9.3 Acceptance and Flight Assurance Tests Performance	9-7
9.3.1 Initial Acceptance Test	9-7
9.3.2 Flight Assurance Tests	9-7
9.3.3 System Acceptance Test	9-8
9.4 Software and Systems-Level Test Facilities	9-9
9.4.1 Digital Computer Emulation	9-9
9.4.2 Software Development Facility	9-10
9.4.3 Avionics Integration Support Facility	9-10
9.4.4 Iron Bird	9-10
9.4.5 Flight System in Aircraft	9-12
9.5 Software Verification	9-13
9.5.1 Quality Assurance in Design	9-13
9.5.2 Programming	9-14
9.5.3 Module Verification	9-15
9.5.4 Module Integration on Flight Computer	9-15
9.6 System Validation	9-16
9.6.1 Independent Verification and Validation	9-16
9.7 Piloted-Mission-Profile Testing	9-25
9.7.1 Aircraft Integration Testing	9-26
9.8 System Flight Test	9-27
9.8.1 High-Speed Taxi Tests	9-27
9.8.2 Flight Tests	9-28
9.9 Software Anomaly Experience	9-28

9.9.1 Software Modification Experience	9-29
9.10 Synopsis of Current Digital Validation Experience	9-30
9.11 References	9-31

LIST OF TABLES

Table	Page
9-1 FDIR Test Procedure (Reference 1)	9-17
9-2 FDIR Test Matrix (Reference 1)	9-19
9-3 Examples of Simulated Software Faults (Reference 1)	9-20
9-4 Summary of FDIR Test Results (Reference 1)	9-20
9-5 Stress Test Procedures (Reference 1)	9-22
9-6 Stress Test Results (Reference 1)	9-23
9-7 Piloted FMET Series (Reference 1)	9-24
9-8 Results of Piloted FMET Series (Reference 1)	9-25
9-9 Aircraft Integration Tests (Reference 1)	9-27
9.10 In-flight Computer and IFU Failure Experience (Reference 1)	9-28
9-11 Software Anomaly Experience (Reference 1)	9-29
SECTION 10	10-1
10. RECOMMENDED VALIDATION PROCEDURES	10-1
10.1 Overview	10-1
10.2 Advanced Digital Integrated Flight Control and Avionics Validation Methodology	10-1
10-2.1 Concept	10-10
10.2.2 System Definition Phase Activities	10-10
10.2.3 System Design Phase Activities	10-18
10.2.4 System Full-Scale Development Activities	10-24
10.2.5 System Integration/Test Activities	10-25
10.2.6 Production and Deployment Activities	10-28
10.2.7 Operation and Maintenance Activities	10-28
10.3 References	10-30

LIST OF ILLUSTRATIONS

Figure		Page
10-1	Systems Design/Development Process (Industry/FAA)	10-2
10-2	Expansion of FAA Interest/Viewpoint	10-2
10-3	Software Life Cycle Activities Time Relationship	10-3
10-4	Digital Systems Validation Activities Sequence	10-4

LIST OF TABLES

Table		Page
10-1	Activities and Documentation	10-7
10-2	Seven Steps of Systems Engineering (Reference 1)	10-10
SECTION 11		11-1
11.	RECOMMENDED CONFIGURATION MANAGEMENT PROCEDURES	11-1
11.1	Configuration Management Plan	11-1
11.1.1	Configuration Identification	11-3
11.1.2	Configuration Control Board	11-4
11.1.3	Configuration Status Accounting	11-6
11.2	Software Configuration Management	11-7
11.3	Summary	11-8
11.4	References	11-9

GLOSSARY

This glossary contains definitions of terms used but not defined in the text of this handbook and of terms likely to be encountered during validation of digital systems. While the glossary is as comprehensive as possible, no claim is made that it is all-encompassing.

Accelerated Stress Testing	Testing in which the applied stress level is chosen to exceed that stated in the reference conditions in order to shorten the time required to observe the stress response of the item or magnify the response in a given time.
Acceptance	The determination by the user (customer) that the product meets his requirements.
Acceptance Test Procedure (ATP)	The configuration controlled, explicitly defined sequence of tests to which the system is subjected for purposes of establishing its acceptability for entry into operational service.
Active Control System	A system which actively commands the movement of control surfaces on the basis of sensor inputs to provide some function or characteristic not available in the aircraft passively.
Algorithm	An explicit set of rules, generally mathematical in nature, for solving a particular problem. When this set of rules is applied to identified inputs, the desired outputs will be obtained after a finite number of steps have been completed.
Allocated Memory	Allocated memory is the sum of that required for the computer programs and associated data base to perform: (1) The baseline functions, plus (2) growth or provisional functions.
Application Software	The part of the operational software which performs specific functions; e.g., navigation computations, sine computations, etc.
Assembler	A program which translates source code statements in mnemonic assembly language instructions into the binary instructions used by the processors, assigns values to named addresses, and performs other functions as an aid to the programmer in writing a software program.
Assembly Language	A programming language which uses the set of processor executable instructions in mnemonic format to write the software program.

Assertion Checking	Evaluating a program by embedding statements that should always hold true.
Assurance Analysis	Is conducted to ascertain that the program performs all of its intended functions and does not perform unintended functions that could degrade or compromise the safety or security of the system to which it belongs.
Auditing	Examinations of software and its documentation for consistency and traceability.
Autopilot	Equipment which automatically performs functions that were normally performed by a pilot, such as maintaining heading or altitude.
Availability	Probability that an item is in the operable and dispatchable state at the start of the mission.
Background Processing	The processing of lower priority functions, which may be interrupted in order for the computer to process higher priority functions assigned as foreground processing.
Baseline	<p>The documented, approved description of the system at any point in time. Baseline items are distinguished from exercises, which represent nonapproved or not-yet-approved trade study items, candidate change items, or recommended items. Depending upon the complexity of the system being developed, the baseline may be described in several distinct steps. Upon approval of each step, which may be marked by a development program milestone, the baseline may be replaced under formal change control. Typical baselines are:</p> <p><u>The Functional or Requirements Baseline</u> is contained in the System Requirements Document and approved at program go-ahead or at the Requirements Review.</p> <p><u>The Allocated Baseline</u> details the allocation of requirements on the software and is contained in the Software Requirements Document. Approval may be at the Requirements Review or the Preliminary Design Review.</p> <p><u>The Design Baseline</u> details the technical design description and is contained in the Design Description Document. Approval is at the Critical Design Review.</p>

The Product Baseline describes the developed product at the time of delivery by the developer to the installer and is contained in the Unit Configuration Index Document. Approval may be at the Functional Configuration Audit.

The Operational Baseline is the product baseline as periodically updated during the operations and maintenance phase of the system life cycle. It is documented in updates to the Unit Configuration Index.

Built-In Test (BIT)

Test equipment, integrated into subsystem, which is used to detect and isolate faults in line replaceable units (LRU). Periodically activated test sequences, generated by software or by hard-wired commands, to detect malfunctions during system operation, which may (or may not) employ Built-In Test Equipment (BITE).

**Built-In Test
Equipment (BITE)
Software**

The portion of the Operational Software which performs test and failure annunciation primarily for the assistance of line maintenance personnel. Built-In Test (BIT) is performed (using BITE, or not) to periodically test hardware components and software initiated functions during operational phases — diagnostics are generally used to assist maintenance personnel previous to system operation.

**Central
Processing Unit (CPU)**

The part of a computer that controls the interpretation and execution of instructions.

Certification

The process of obtaining regulatory agency approval for a function, equipment, system, or aircraft, by establishing that it complies with all applicable government regulations.

Change Control

The process of evaluating, approving, and documenting changes to the system.

Code

The representation of particular data or a particular computer program in a symbolic form, such as source code, object code, or machine code.

**Command Augmentation
System (CAS)**

An active control system that augments the pilot's control inputs with sensor inputs to provide him direct control of aircraft motion rather than control surface position.

Compiler	Software that translates source code statements in a high order language, such as FORTRAN or PASCAL, into assembly language or object code for a particular computer.
Complexity	A formal measure of a program's difficulty in terms of algorithm, constructs, and data flow, but not in terms of length.
Consistency	Uniformity of notation, terminology, and symbology within a program, and its modules, which should be arranged in a uniform architecture.
Control Configured Vehicle (CCV)	An aircraft whose basic aerodynamic and/or structural design can include the use of an active control system.
Control Law	A set of equations which define control surface position as a function of sensed inputs.
Coverage	The conditional probability that given the existence of a fault in an operational system, the system is able to recover and continue operation with no permanent loss of function.
Critical Design Review (CDR)	A review to verify the adequacy of the design to satisfy the requirements in the System Requirements Document and the Software Requirements Document; to establish a firm design baseline; to assess risk areas; and to approve commencement of qualification, verification, and validation activities. Documentation requirements may include: Design Description Document(s), engineering analyses, and plans and procedures.
Correctness Proof	Technique of proving mathematically that a given program is correct with a given set of specifications. The process can be accomplished by manual methods or by program verifiers requiring manual interactions.
Cross Assembler	An assembler which executes on one computer and generates machine code for a different computer.
Cross Channel Monitoring	The process by which the signals or outputs of the channels are compared and any disagreement, outside of a tolerance range, is classified as a fault.
Cross-Strapping	The physical hardwiring of an element in one channel to elements in other channels.

Data	Inputs in the form of a binary string that may have significance beyond their numerical meaning.
Data Base Software	Software that contains the numerical values of the parameters required for program computations to be stored in computer memory.
Data-Flow Analysis	Graphical analysis of sequential data patterns; e.g., by tools that identify undefined variables.
Debug	The development process to locate, identify, and correct programming mistakes, including omissions, from software.
Decomposition	Breaking down a software specification, in depth and breadth, to determine all required functions and their relationships.
Diagnostics	An output from a tool, indicating software discrepancies and other attributes.
Digital Computer	A system containing a processor, variable storage memory, program storage memory, input and output interface circuits, and support circuits including control, timing, power supply, etc. The computer can perform a large variety of functions by the sequential execution of a set of basic operations in the processor. The commands for the set of operations is called the software program and is stored in the program memory. (The hardware necessary to convert input signals to the proper digital form and also the hardware necessary to convert the output signals to the proper form is usually included within the definition of a computer.)
Distributed System	A system where the functions are distributed to a number of different computers.
Dynamic Analysis	Execution of an instrumented program to collect information on its behavior and correctness.
Elastic Mode Suppression (EMS)	Active control to increase the damping of lightly damped structural bending modes excited by gusts.
Electrical Command System	A system in which electrical signals provide the primary control commands but a mechanical backup system is retained.

Emulator	Software run on a host computer that accepts the same input data, executes the same programs, and yields the same outputs as the target computer. The emulation software may execute on a host computer or on a computer similar to the computer that will actually be used in the system. Emulators replace the computer in the system to enable the computer/system interface to be tested, verified, and validated in an orderly fashion.
Error	Variation in measurements, calculations, or observations of a quantity due to mistakes or uncontrollable factors.
Fail-Operational	A system which is able to continue to provide critical functions after one failure.
Fail-Passive	A system that does not cause an unsafe condition after a failure. There is no disruption in the aircraft. The function just ceases to be performed.
Fail-Safe	A system that does not cause an unsafe condition after a failure. Immediate pilot corrective action may be necessary.
Fail-Soft	A system that does not cause an unsafe condition after a failure. Pilot corrective action may be required within, for example, six seconds.
Failure	Omission of occurrence or performance; specifically, a failing to perform a duty or expected action. A condition which can give rise to a fault, usually considered permanent.
Fault	An anomaly in the performance of a digital system due to an unspecified and disruptive change in one or more logic variables.
Fault Tolerant	<p>The ability of the system to experience a finite number of failures and continue operation, in either a fully operational or degraded mode. A fault tolerant system may be considered to be a system which provides the correct execution of a function at all times and encompasses:</p> <ol style="list-style-type: none"> (1) Elimination of hardware design errors; (2) Correctness and completeness of software specifications;

- (3) Testing, verification, and validation of programs and microprograms;
- (4) Continued correct execution of programs in the presence of hardware (physical) faults.

Federated System

A system where different functions are performed by different computers and all are connected together into a total system.

First Article Inspection

Verifies that the "as-built" unit conforms to the corresponding technical documentation. Performed on production items representing first-off-the-line configurations.

Flight Safety Reliability

The probability, per flight, of not losing the aircraft due to failures in the engine, flight control, avionics, electrical system, or crew.

Flutter Suppression

Active control to suppress aeroelastic flutter modes.

Fly-By-Wire (limited definition)

The use of electrical signals to connect the pilot's control devices with the control surfaces.

Fly-By-Wire (broader definition)

The use of electrical control connections with no mechanical backup linkages and providing the pilot direct control of aircraft motion rather than control surface position.

Foreground Processing

The processing of time-critical, noninterruptible real-time functions within the computer processor, as contrasted to background processing.

Function

Each special purpose operation or action performed by a system, subsystem, unit or part, that is required to conduct the mission.

Critical Functions

Are those for which the occurrence of any failure condition or design error would prevent the continued safe flight and landing of the aircraft.

Essential Functions

Are those for which the occurrence of any failure condition or design error would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions.

Non-Essential Functions

Are those for which failure conditions or design errors could not significantly degrade aircraft capability or crew ability.

Functional Configuration Audit (FCA)	A procedure to establish that the functional characteristics of the system software fulfill the development specification requirements.
Gust Load Alleviation (GLA)	Active control to reduce loads due to gust.
High Order Language (HOL) (High Level Language)	A type of source language which is problem- or function-oriented that enables code to be written in a more readily understandable form than object code and can be automatically translated into object code. Most HOLs, such as FORTRAN or PASCAL, are not restricted to application on only one type of computer.
Host Computer	Any computer used to develop software for another (target) computer.
In-Line Channel Monitoring	The process by which the signals or outputs of a single channel are checked (for faults) by the processor of the channel.
Installer	The group or organization responsible for product installation design, system integration, and certification demonstration of a product on the aircraft: Usually the aircraft manufacturer, although it may be a separate installation contractor, or sometimes an element of the user's organization.
Instrumentation	Adding code to a program for injecting data and collecting information, usually for dynamic analysis.
Integrated System	A system in which the design has been integrated to allow the optimum synergistic arrangement for the performance of functions and the use of resources.
Interface Analysis	Checking the interfaces between program elements (modules) for consistency and proper data transfer.
Intermediate Code	Machine input in a form between source and machine code, for example, pseudocode.
Line Replacement Unit (LRU)	The smallest element of a system normally removed and replaced on the aircraft while in operational status by the line maintenance crew.

Linkage Editor (Linker)	Software that combines separately produced blocks of object code produced by assembler or compiler; resolves symbolic cross references among them; replaces, deletes and adds control sections, and produces executable code.
Loader	Software that accepts load modules created by the linkage editor, and loads executable code directly into main storage.
Machine Code (Machine Language)	The binary form of computer instructions, which is directly executable by the CPU. This is the lowest level language in which programs may be written.
Man-Made Faults	Are divided into (1) development faults which occur during the development phase; i.e., incomplete, ambiguous, or erroneous specifications and (2) interaction faults which are faults caused by inputs that are introduced into the system via the man-machine interfaces during operation or maintenance phases by the user.
Mission Reliability	The probability that the device will successfully complete its defined mission.
Module	A uniquely identified element of a computer program which performs a specific function or set of related functions.
Non-Volatile Memory	Memory which does not require power to retain the stored data.
Object Code	Object code is a low-level representation of the computer program that may be in a directly usable form, such as machine code or in a form which incorporates relocation information in addition to the instruction information.
Operational Software	Operational software (or flight software or resident software) is all software resident in the system and in use while installed in its operating environment. It includes executive software, functional (or applications) software data base software, and BITE software.
Partitioning	The process of determining how the system requirements will be implemented either in hardware and its components or in software and its components. In software, partitioning is said to exist if co-resident tasks execute without any interdependency between them.

Physical Fault	Caused by a physical failure phenomena that could effect one or more components of the system and cause either a permanent or temporary change to the values of the physical variable.
Preliminary Design Review (PDR)	A review to determine compatibility of the selected design approach with the performance and functional requirements of the System Requirements Document, to formalize the Allocation Baseline, and to obtain approval for commencement of the detailed design phase.
Probes	Statements used for program "instrumentation."
Processor	Electronic circuits capable of sequentially performing a set of basic arithmetic, logical, and control operations. A processor is the central element of a computer.
Program Stubs	Code sections added to a subprogram to make it executable; stubs are usually substitutes for parts of the missing main program.
Qualification	Entails ensuring that a product meets or exceeds a minimum industrial and/or commercially accepted standard.
Quality Assurance	A systematic pattern of actions throughout design and production, to assure confidence in a product's conformance with specifications.
Recovery	Comprises all actions that are initiated by the arrival of a fault signal during normal operation and are concluded by the resumption of normal operation (possibly in a degraded mode) by systematic shutdown of the system, or by system failure.
Redundancy Management	The process of managing redundant elements in order to identify a failure and then reconfiguring the system to remove the effects of the failed element and continue operation with unfailed elements.
Regression Testing	Method for detecting errors spawned by corrections during software development and maintenance.
Relaxed Static Stability (RSS)	The use of active control to allow the static stability of the basic unaugmented airframe to be relaxed. The aircraft with the active system operating will have the normal stability margins.

Ride-Control System (RCS)	Active Control to improve the quality of the ride for the crew and passengers.
Robustness	The ability of a program to withstand stresses (input quantity and quality) beyond the range for which it was designed.
Simulation	Representation of either an abstract or a physical system's features by computer operations. Often the operating environment of a program must be simulated during software testing.
Simulator	Software run on a host computer for use in system function testing. It uses the input data and provides the output data defined for the system operational software to be run on the target computer. It is not intended to emulate the transfer characteristics of the target computer. In the broader sense, a simulator is any device or system which generates artificial conditions for test or training purposes.
Sizing	Estimate or measurement of the amount of memory required or actually utilized, especially as compared with total memory available.
Software	Computer programs.
Software Program	The set of processor-executed instructions stored in the memory of a computer which cause the computer to perform the desired functions.
Software Problem Report (SPR)	A report of a perceived problem that the software does not meet the defined requirements or perform intended functions.
Source Code	Code that is developed using the source language (assembly language or HOL).
Stability Augmentation System (SAS)	An active control system which augments the natural stability of an aircraft.
Standards	Specifications that refer to the method of development procedures, rules, conventions, and guidelines used for prescribing all or any part of program design, coding, and testing.

Static Analysis	Examination of a program (usually via computer) for errors and inconsistencies, without actual execution.
Status Accounting	The process of documenting the current approved status of the system including an historical record of all approved changes.
Support Software	All software that is used in the development, verification, validation, and modification of the operational software. Support software includes: Compilers, assemblers, emulators, simulators, editors, linking loaders, debugging programs, operator training programs, and document generation and control programs.
Symbolic Execution	Reconstructing the logic and actions along a program path via symbolic rather than actual data.
Synthesis	The substitution of data calculated from the physical relationships of the system using other parameters for a failed element.
Target Computer	The digital computer embedded in the operational equipment that executes the operational software.
Test Bed	A test site that either contains or simulates all hardware and software interfaces.
Test Driver	Tool providing the facilities needed to execute a program; e.g., inputs or files and commands. May also evaluate outputs and produce reports.
Testing	The process of executing a program with the intent of finding errors.
Throughput	A measure of the computing capability of a processor, normally expressed in thousands of operations per second (KOPS) of a specified instruction mix.
Timing	Estimate or measurement of the amount of computation time required or actually utilized. Usually compared with the available time for computation to determine timing margin.
Traceability	The ability to trace or associate each function between the various documentation levels which define the system and software requirements, test and design. Symbol identification

	that permits symbols to be traced throughout a software system. Also, continuity between program versions (see Audit).
Tracer Program	Analysis tool that searches programs for unreachable ("dead") code.
Transient Fault	A temporary anomaly in the performance of a system.
User (End User)	The group or organization using the system containing the delivered software on an operational basis; the airplane operator, the ultimate customer of the developer or installer.
Validation	The process of demonstrating, through testing in the real environment, or an environment as real as possible, that the system not only is verifiable but also satisfies the user's requirements, with no undesired effects.
Verification	The process of demonstrating the logical correctness and showing that the product performs according to its specification which should reflect the system requirements and software requirements.
VHLL	Very-high-level language, usually a problem or requirements-description language ranging in form from the highly abstract to plain English.
Volatile Memory	Memory that requires a continuous supply of power applied to its internal circuitry to prevent the loss of stored data.

SECTION ONE
INTRODUCTION

TABLE OF CONTENTS

	Page
SECTION 1	1-1
1. INTRODUCTION	1-1
1.1 Purpose	1-1
1.2 Distribution	1-1
1.3 Background	1-2
1.4 Handbook Scope	1-7
1.5 Handbook Organization	1-8
1.6 References	1-10

SECTION 1

1. INTRODUCTION

1.1 PURPOSE.

The purpose of this handbook is to identify techniques, methodologies, tools, and procedures in a systems context that may be applicable to aspects of the validation and certification of digital systems at specific times in the development, and implementation of software based digital systems to be used in flight control/avionics applications. The application of these techniques in the development of discrete units and/or systems will result in completion of a product or system which is verifiable and can be validated in the context of the existing regulations/orders of the government regulatory agencies. The handbook uses a systems engineering approach to the implementation and testing of software and hardware during the design, development, and implementation phases. The handbook also recognizes and provides for the evaluation of the pilot's workload in the utilization of the new control/display technology, especially when crew recognition and intervention may be necessary to cope with/recover from the effects of faults or failures in the digital systems or the crew introduces errors into the system under periods of high workload due to some inadvertent procedure or entry of incorrect or erroneous data.

In summary, the handbook:

1. Identifies and presents the issues related to the design, development, and implementation of software based digital systems.

2. Identifies specific approaches applicable to all aspects of the verification and validation procedures, at specific times in the development and certification portion of the system life cycle.

3. Provides the government regulatory agencies (especially the Federal Aviation Administration (FAA)) as well as industry with a set of tools/procedures, in a systems engineering context which may be of value in the validation/certification process.

This handbook was developed in support of ongoing activities of the Radio Technical Commission (RTCA), Society of Automotive Engineers (SAE), Aeronautical Radio (ARINC), National Aeronautics and Space Administration (NASA), Department of Defense (DOD) agencies (U.S. Air Force (USAF), U.S. Navy (USN), U.S. Army (USA), and various industry groups and companies.

1.2 DISTRIBUTION.

It is intended that this validation handbook will be distributed to the certification engineers responsible for systems certification in the four certification directorates (Northwest Mountain Region (ANM), Central Region (ACE), Southwest Region (ASW), and New England Region (ANE) within the FAA as well as systems engineers within the Associate Administrator for Aviation Safety (AVS), Office of Airworthiness (AWS), Office of Aviation Safety (ASF), the Office of the National Resource Specialist for Digital Systems (ANM-103N), and the Aircraft Certification Offices.

The handbook will also be available to Airways Facilities, Systems Research and Development Service, Air Traffic Control Service, and Flight Standards elements.

Even though the handbook uses flight control/avionics systems for examples, the materials in this handbook is applicable to any component or systems design, development, and implementation activity within the FAA and other government agencies which requires software based digital systems for mechanization and implementation.

It is intended that this handbook will also be available to industry groups and companies and will provide a possible means of complying with regulations, orders, and advisory materials associated with certification requirements.

1.3 BACKGROUND.

a. Historical

Certification of digital flight control and avionics systems (DFCAS) has been receiving a great deal of attention within the FAA and industry since 1975 (references 1-14). In the summer of 1975, the FAA and NASA began planning a joint program aimed at assessing and selectively upgrading the assurance technology applicable to digital flight control and avionic systems (reference 1). Their respective interests and those of industry, as noted in reference 1, were then incorporated into a research strategy. Note that the common goal is the attainment of practical verification and validation assurance technology advances. Currently, the implementation of this strategy has progressed to the point of providing significant returns on the research and technology (R&T) investment.

In April 1976, the FAA and NASA cosponsored a government/industry workshop in digital flight controls and avionics in anticipation of the widespread introduction of such systems. The FAA then expressed its intent to avoid delays in the certification process, inordinate costs to industry, and any adverse effects upon safety (reference 3). They indicated particular interest in system simulation methods and pilot roles under faulted operation. At the same time, a considerable body of industry recommendations were compiled, analyzed, and used in the formulation of the validation technology R&T strategy within the FAA. Many of these recommendations, moreover, remain important considerations in the FAA's continuing efforts.

In December 1976, the FAA instituted the Advanced Integrated Flight Systems (AIFS) Program to evaluate and advance the aviation safety regulatory system and policies relative to emerging advanced technologies (reference 4). Specifically oriented towards digital technology, the following areas were highlighted: Simulation for validation; fault tolerance for flight-critical applications (especially active controls); reliability and safety assessment methods; and lightning effects. Rather significantly, these still remain the highest priority R&T areas within the FAA's research and development (R&D) and certification directorates.

At the time of the formulation of the AIFS Program, these technology areas were purposefully oriented towards the FAA's regulatory organization needs. References 4 and 5, for example, emphasize the necessity of research support and awareness of regulatory personnel. More than ever, this constitutes a basic commitment of the FAA.

To maintain a purposeful sense of direction in addressing the foregoing technology problems, the FAA is closely attuned to the outputs of certain government and/or industry organizations. Resultant documents of particular significance are the following:

RTCA's DO-178, "Software Considerations in Airborne Systems and Equipment Certification" (Reference 17)

FAA's Advisory Circular (AC) 25.1309-1, "FAR Guidance Material, Airplane System Design Analysis" (Reference 18)

SAE's S-18A Committee Draft ARP 926B, "Fault/Failure Analysis Procedure for Hardware/Systems" (Reference 19)

In developing a practical methodology then, an added dimension is that of accommodating or anticipating the needs reflected in such documents. In this vein, the FAA recognizes that there is not nor will there be, just one methodology which can fulfill flight-critical assurance needs. The subject handbook is merely seeking to demonstrate at least one representative and effective way of assuring the safety of flight-critical DFCS by providing a summary of the methodologies, techniques, and tools (with many workload examples) which should be used in order to provide a plan and a product which will comply with the FAA's orders, regulations, and advisory materials specified (or called out) in the certification process.

b. Regulations, Orders, and Advisory Circulars

The FAA has established certain ground rules, procedures, and guidance material which describe the requirements that aircraft, and systems within the aircraft, must meet for certification. Examples of these documents include Order 8110.4, Order 8110.8, AC 25:1309-1, and AC 120-28C. In addition, the FAA has established airworthiness standards, especially Part 25, Airworthiness Standards: Transport Category Airplane (reference 20). These standards specifically require that the airplane (or the systems within the airplane) must be shown by analysis, test, and documentation to meet the specific requirements contained in the subject document.

The specific sections within FAR 25 relevant to digital flight control and avionics systems are:

- | | |
|-----------------|---|
| 1. FAR 25:671 | Control Systems |
| 2. FAR 25:672 | Stability Augmentation and Automatic and Power-Operated Systems |
| 3. FAR 25:673 | Two-Controlled Airplanes |
| 4. FAR 25:1301 | Function and Installation |
| 5. FAR 25:1303 | Flight and Navigation Instruments |
| 6. FAR 25:1309 | Equipment, Systems, and Installation |
| 7. FAR 25:1321 | Arrangement and Visibility |
| 8. FAR 25:1329 | Automatic Pilot System |
| 9. FAR 25:1331 | Instruments using a Power Supply |
| 10. FAR 25:1333 | Instruments Systems |
| 11. FAR 25:1335 | Flight Director Systems |
| 12. FAR 25:1351 | Electrical Systems and Equipment |

The above FAR's are extracted and presented below. These FAR's as evidenced by the standards have their requirements stated on a subsystem by subsystem basis (e.g., control system, stability augmentation system, electrical system, and equipment) with only sections such as "Equipment Systems and Installation (25.1309)" that generally cover the interaction of all subsystems. The military specification for flight controls systems (MIL-S-9490D) follows a similar arrangement and specifically excludes crew displays and electronics not dedicated to flight control. As a result, various interpretations have been developed as to the certification requirements and the appropriate procedure (analysis, test, simulation, etc.) to be applied to a system to demonstrate that it complies with the certification standards.

"Control Systems (25.671)

- (c) The airplane must be shown by analysis, test, or both, to be capable of continued safe flight and landing after any of the following failures . . .
 - (1) Any single failure, excluding jamming.
 - (2) Any combinations of failure not shown to be extremely improbable, excluding jamming (for example, dual electrical/hydraulic system failures, or any single failure in combination with any probable hydraulic or electrical failure).
 - (3) Any jam in a control position normally encountered during takeoff, climb, cruise, normal turns, descent and landing unless the jam is shown to be extremely improbable, or can be alleviated. A runaway of a flight control to an adverse position and jam must be accounted for if such runaway and subsequent jamming is not extremely improbable.
- (d) The airplane must be designed so that it is controllable if all engines fail. Compliance with this requirement may be shown by analysis where that method has been shown to be reliable."

"Stability Augmentation and Automatic and Power-Operated Systems (25.672)

If the functioning of stability augmentation or other automatic or power-operated systems is necessary to show compliance with the flight characteristics requirements of this part, such systems must comply with paragraph 25.671 and the following:

- (b) The design of the stability augmentation system or of any other automatic or power-operated system must permit initial counteraction of failures of the types specified in paragraph 25.671(c) without requiring exceptional pilot skill or strength, by either the deactivation of the system, or a failed portion thereof, or by overriding the failure by movement of the flight controls in the normal sense.
- (c) It must be shown that after any single failure of the stability augmentation system or any other automatic or power-operating system:

- (1) The airplane is safely controllable when the failure or malfunction occurs at any speed or altitude within the approved operating limitations that is critical for the type or failure being considered;
- (2) The controllability and maneuverability requirements of this part are met within a practical operational flight envelope . . .
- (3) The trim, stability, and stall characteristics are not impaired below a level needed to permit continued safe flight and landing."

"Two-Controlled Airplanes (25.673)

Two-controlled airplanes must be able to continue safely in flight and landing if any one connecting element in the directional-lateral flight control system fails."

"Equipment Systems and Installation (25.1309)

- (b) The airplane system and associated components, considered separately and in relation to other systems, must be designed so that
 - (1) The occurrence of any failure condition which would prevent the continued safe flight and landing of the airplane is extremely improbable.
 - (2) The occurrence of any other failure conditions which would result in injury to the occupants or reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions is improbable.
- (d) Compliance with the requirements of paragraphs (b) and (c) of this section must be shown by analysis, and where necessary, by appropriate ground, flight, or flight simulated tests. The analysis must consider
 - (1) Possible modes of failure, including malfunctions and damage from external sources.
 - (2) The probability of multiple failures and undetected failures.
 - (3) The resulting effects on the airplane and occupants, considering the stage of flight and operating conditions and
 - (4) Crew warning cues, corrective action required, and the capability of detecting faults.
- (e) Each installation whose functioning is required by this subchapter, and that requires a power supply, is an "essential load" on the power supply. The power sources and the system must be able to supply the following power loads in probable operating combinations and for probable duration:

- (1) Loads connected to the system with the system functioning normally.
- (2) Essential loads, after failure of any one prime mover, power converter, or energy storage device.
- (3) Essential loads after failure of
 - (i) Any one engine on two- or three-engine airplanes; and
 - (ii) Any two engines on four- or more-engine airplanes
- (4) Essential loads for which an alternate source of power is required by this chapter, after any failure or malfunction in any one power supply system, distribution system or other utilization system.
- (f) In determining compliance with subparagraphs (e)(2) and (3) of this section, the power loads may be assumed to be reduced under a monitoring procedure consistent with safety and the kinds of operation authorized. Loads not required in controlled flight need not be considered for the two-engine-inoperative condition on airplanes with four or more engines."

"Automatic Pilot System (25.1329)

- (a) Each automatic pilot system must be approved and must be designed so the automatic pilot can be quickly and positively disengaged by the pilots to prevent it from interfering with their control of the airplane.
- (f) The system must be designed and adjusted so that, within the range of adjustment available to the human pilot, it cannot produce hazardous loads on the airplane, or create hazardous deviations in the flightpath, under any condition of flight appropriate to its use, either during normal operation, or in the event of a malfunction, assuming that corrective action begins within a reasonable period of time.
- (g) If the automatic flight integrates signals from auxiliary controls or furnishes signals for operation of other equipment, there must be positive interlocks and sequencing of engagement to prevent improper operation. Protection against adverse interaction of integrated components, resulting from a malfunction, is also required."

"Instrument Systems (25.1333)

For systems that operate the instruments required by paragraph 25.1303(b) which are located at each pilot's station:

- (b) The equipment, systems, and installation must be designed so that one display of the information essential to the safety of flight which is provided by the instruments, including attitude, direction, airspeed, and altitude will remain available to the pilots, without additional crew member

action, after any single failure or combination of failures that is not shown to be extremely improbable; and

- (c) Additional instruments, systems, or equipment may not be connected to the operating system for the required instruments, unless provisions are made to insure the continued normal functioning of the required instruments in the event of any malfunction of the additional instruments, systems, or equipment which is not shown to be extremely improbable."

"Electrical Systems and Equipment (25.1351)

- (b) Generating system. The generating system includes electrical power sources, main power busses, transmission cables, associated control, regulation, and protective devices. It must be designed so that:
 - (1) Power sources function properly when independent and when connected in combination;
 - (2) No failure or malfunction of any power source can create a hazard or impair the ability of remaining sources to supply essential loads."

1.4 HANDBOOK SCOPE.

The scope of this handbook is to specify techniques, methodologies, tools, and verification/validation strategies for the organized design, development, and implementation of software based digital systems by an applicant seeking FAA certification under the procedures described in Order 8110.4, the relevant FAR's, and advisory circulars (especially 25:1309-1).

The techniques, methodologies, tools, and verification and validation strategies are meant to be incorporated into an overall plan which should, as a minimum, comply with Section 3.0 of RTCA DO-178, RTCA DO-160A, relevant TSO documents, or with ATA 102 or any other DOD document which specifies the requirements for criticality assessment, software development, hardware development, configuration management, validation and verification in the context of flight control and avionics systems.

The key words (and/or areas of consideration) which should be addressed in any plan developed in accordance with the above objectives are:

Validation	Verification
Failure Modes and Effects	Fault Tree
Reliability	Maintainability
Fault Detection/Isolation	Fault Insertion
Emulation	Simulation
Safety Assessment	Configuration Management/Control

System Design	Built-in Test
Monitoring (Schemes)	Redundancy Management
System Recovery	Documentation
Systems Certification	Life Cycle

If these areas are not addressed (as a minimum), then it may be inferred that the system, under consideration, has not been developed in accordance with the handbook's terms of reference or the relevant orders, FAR's, and advisory documents issued by the FAA in support of system certification. Conversely, the FAA's certification engineers can use this handbook as background and guidance material to support the evaluation of data packages submitted by an applicant during the certification process.

1.5 HANDBOOK ORGANIZATION.

This document is organized in eleven sections plus supporting appendices.

Section 2, Applicable Documents, lists the principal documents referred to by this handbook. Each of the sections contains a list of references for that section.

Section 3, System Life Cycle, describes the system development process by phase and identifies the products of each of the phases relevant to the validation of digital systems.

Section 4, Mission Factors, discusses important mission factors which must be considered in the validation of digital systems. The advanced digital integrated flight control and avionics systems functions are also described.

Section 5, System Architectures, synthesizes pertinent ARINC characteristics, and discusses software, system monitors, and system configurations.

Section 6, Crew Workload Evaluation/Man-Machine Interface, provides a summary of workload evaluation methods and procedures for evaluation of the man-machine interface.

Section 7, Issues, contains a discussion of various issues related to the validation of digital systems.

Section 8, Concepts and Methodologies, presents a synopsis of various methods proposed or used for validation of digital systems.

Section 9, Current Validation Procedures, describes design validation, hardware testing, acceptance and flight assurance tests, software and system-level test facilities, software verification, system validation, the pilot's role in validation, and system flight test experience.

Section 10, Recommended Validation Procedures, describes the procedures and the sequence of their application for validation of digital flight control and avionics systems.

Section 11, Recommended Configuration Management Procedures, discusses procedures for configuration management before and after validation and certification of the digital system.

Appendix A presents a recommended specification format for documentation including system level and software.

Appendix B presents a synopsis of reliability analysis models, fault trees, and failure modes and effects analysis.

Appendix C provides an example of a fault tree application.

Appendix D contains advisory circular AC: 25:1309-1.

Appendix E contains advisory circular AC: 20-115.

SECTION 1

1.6 REFERENCES

1. Government/Industry Workshop on Methods for the Certification of Digital Flight Controls and Avionics, NASA TMX-73, 1/4, October 1976.
2. Validation Methods for Fault-Tolerant Avionics and Control Systems-Working Group Meeting I, NASA Conference Publication 2114, March 12-14, 1979.
3. Validation Methods Research for Fault-Tolerant Avionics and Control Systems--Working Group Meeting II, NASA Conference Publication 2130, October 3-4, 1979.
4. Boothe, E. M., et al, Engineering and Development Program Plan - Advanced Integrated Flight Systems (AIFS), Report No. FAA-ED-18-3, ARD-530, Systems Research and Development Service, Federal Aviation Administration, Department of Transportation, Washington, D.C., May 1978.
5. Traybar, J. J. et al, Engineering and Development Program Plan-Advanced Integrated Flight Systems (AIFS), Report No. FAA-ED-18-3A, Federal Aviation Administration Technical Center, Atlantic City Airport, N.J., September 1981.
6. Mulcare, Dennis B., and Rang, Edward R., Comprehensive Methodologies for Digital Flight Control Software Verification, NAECON '80 Record, pp. 252-259.
7. Wattenberg, Robert E., Independent Test and Evaluation, Proceedings of the Aeronautical Systems Software Workshop, 1974, pp. 333-338.
8. Evans, Michael, Verification and Validation, IBID, pp. 339-343.
9. Roderick, Theodore L., Verification and Validation of Aeronautical Systems Software, IBID, pp. 345-349.
10. Ziemer, Donald A., and Macy, Spencer M., An Approach to Verification and Validation of Operational Software, IBID, pp. 350-354.
11. Rubey, Raymond J., New Approaches for Software Validation, NAECON '72 Record, pp. 252-258.
12. Waterman, Hugh E., FAA's Certification Position on Advanced Avionics, Astronautics and Aeronautics, May 1978, pp. 49-51.
13. Mulcare, D. B., Ness, W. G., McCarty, J. M., et al, Industry Perspective on Simulation Methods and Research for Validation and Failure Effects Analysis of Advanced Digital Flight Control/Avionics, Final Report, Contract NAS2-9784, Lockheed-Georgia Company, January 22, 1978.
14. Bridgman, Michael S., Hitt, Ellis F., and Kenney, Suzanne M., Evaluation of Methods for Reliability and Failure Effects Analysis of Advanced Digital Flight Control and Avionics Systems, Battelle-Columbus Laboratories, April 21, 1980.
15. Hitt, Ellis F., Digital Avionics Validation, Final Report DOT-FA03-81-P-00532, Battelle-Columbus Laboratories, February 27, 1981.

16. Bernhard, Robert, The 'No-Downtime' Computer, IEEE Spectrum, September 1980, pp. 33-37.
17. Software Considerations in Airborne Systems and Equipment Certification, RTCA Document No. DO-178, Radio Technical Commission for Aeronautics, September 7, 1982.
18. Airplane System Design Analysis, FAA Advisory Circular 25.1309-1, November 5, 1981.
19. Fault/Failure Analysis Procedure for Hardware/Systems, Society of Automotive Engineers (SAE) Aerospace Recommended Practice (ARP) 926B, SAE S-18A Committee.
20. Federal Aviation Regulations, Part 25, Airworthiness Standards: Transport Category Aircraft, Federal Aviation Administration.

SECTION TWO
APPLICABLE DOCUMENTS

TABLE OF CONTENTS

	<u>Page No.</u>
SECTION 2	
APPLICABLE DOCUMENTS.	2-1

SECTION 2

2. APPLICABLE DOCUMENTS

The following documents of the issue in effect on the date of publication of this handbook form a part of this handbook to the extent specified herein.

U. S. Government Documents

Regulations

Federal Aviation Regulations, Part 25	Air Worthiness Standards-Transport Category Aircraft
Federal Aviation Regulations, Part 29	Air Worthiness Standards-Helicopters
Federal Aviation Regulations, Part 121	Air Carriers and Commercial Operators of Large Aircraft

Advisory Circulars

25.1309-1, September 7, 1982	Airplane System Design Analysis
25.1329-1A, July 8, 1968	Automatic Pilot Systems Approval
120-20, June 6, 1966	Criteria for Approval of Category II Landing Weather Minima
20-57A	Automatic Landing Systems
120-28C, App. 1 & 2, May 13, 1982	Criteria for Approval of Category IIIa Landing Weather Minima, Airworthiness Approval for Category IIIa Airborne Systems
120-29, App.1	Airworthiness Approval for Category II Installations of Airborne Navigation, Instrument, and Flight Control Systems in Transport Category Aircraft

Orders

8110.4, Changes 1 thru 22 October 1978	Type Certification
---	--------------------

Specifications

MIL-F-9490D (USAF), 6 June 1965	Flight Control Systems--Design, Installation and Test of Piloted Aircraft, General Specification for
MIL-F-8785B (ASG), 7 August 1969	Military Specification, Flying Qualities of Piloted Airplanes
MIL-F-83300, December 1970	Military Specification--Flying Qualities of Piloted V/STOL Aircraft
MIL-M-7997	Motor, Aircraft Hydraulic, Constant Displacement, General Specification for
MIL-I-8500	Interchangeability and Replaceability of Component Parts for Aircraft and Missiles
MIL-P-8564	Pneumatic System Components, Aeronautical, General Specification for
MIL-M-8609	Motor, DC, 28 Volt System, Aircraft, General Specification for
MIL-S-8698	Structural Design Requirements, Helicopters
MIL-H-8775	Hydraulic System Components, Aircraft and Missiles, General Specifications for
MIL-A-8860	Airplane Strength and Rigidity, General Specification
MIL-A-8861	Airplane Strength and Rigidity, Flight Loads
MIL-A-8865	Airplane Strength and Rigidity; Miscellaneous Loads
MIL-A-8866	Airplane Strength and Rigidity - Reliability Requirements, Repeated Loads, and Fatigue
MIL-A-8867	Airplane Strength and Rigidity, Ground Tests
MIL-A-8870	Airplane Strength and Rigidity Flutter; Divergence, and Other Aeroelastic Instabilities
MIL-T-8878	Turnbuckle, Positive Safetying
MIL-S-8879	Screw Threads, Controlled Radius Root With Increased Minor Diameter; General Specification for
MIL-H-8890	Hydraulic Components, Type III, -65° to +450°F, General Specification for
MIL-H-8891	Hydraulic Systems, Manned Flight Vehicles, Type III, Design, Installation, and Data Requirements for

Specifications (Continued)

MIL-H-8892	Airplane Strength and Rigidity, Vibration
MIL-A-8893	Airplane Strength and Rigidity, Sonic Fatigue
MIL-B-8976	Bearing, Plain, Self-Aligning, All-Metal
MIL-S-9419	Switch, Toggle, Momentary, Four-Position On, Center Off
MIL-C-18375	Cable, Steel (Corrosion-Resisting, Nonmagnetic) Flexible, Preformed (for Aeronautical Use)
MIL-A-21180	Aluminum-Alloy Casting, High Strength
MIL-A-22771	Aluminum Alloy Forgings, Heat Treated
MIL-K-25049	Knob, Control, Equipment, Aircraft
MIL-G-25561	Grip Assembly, Controller, Aircraft, Type MC-2
MIL-V-27162	Valve, Servocontrol, Electrohydraulic, General Specification for
MIL-C-27500	Cable, Electrical, Shielded and Unshielded, Aircraft and Missile
MIL-E-38453	Environmental Control, Environmental Protection, and Engine Bleed Air Systems, Aircraft, and Aircraft Launched Missiles, General Specification for
MIL-M-38510	Microcircuit, General Specification for
MIL-S-52779	Software Quality Assurance Requirements
MIL-C-81774	Control Panel, Aircraft, General Require- ments for
MIL-B-81820	Bearing, Plain, Self-Lubricating, Self- Aligning, Low Speed
MIL-F-83142	Forging, Titanium Alloys, for Aircraft and Aerospace Applications
MIL-W-83420	Wire Rope, Flexible, for Aircraft Control
MIL-A-83444	Airplane Damage Tolerance Requirements

Standards

Military

MIL-STD-143	Standards and Specifications, Order of Precedence for the Selection of
MIL-STD-203	Aircrew Station Controls and Displays for Fixed Wing Aircraft

Military (Continued)

MIL-STD-250	Aircrew Station Controls and Displays for Rotary Wing Aircraft
MIL-STD-421	Chain Roller; Power Transmission and Conveyor, Flat Link Plates, Single Pitch, Single and Multiple Strand, Connective Links and Attachment Links
MIL-STD-454	Standard General Requirements for Electronic Equipment
MIL-STD-461	Electromagnetic Interference Characteristics Requirements for Equipment
MIL-STD-471A	Maintainability Verification/Demonstration/Evaluation
MIL-STD-480	Configuration Control - Engineering Changes, Deviations and Waivers
MIL-STD-483	Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs
MIL-STD-490	Specification Practices
MIL-STD-499	Engineering Management
MIL-STD-704	Electric Power, Aircraft, Characteristics and Utilization of
MIL-STD-781	Reliability Design Qualification and Production Acceptance Tests: Exponential Distribution
MIL-STD-810	Environmental Test Methods
MIL-STD-838	Lubrication of Military Equipment
MIL-STD-1472	Human Engineering Design Criteria for Military Systems, Equipment and Facilities
MIL-STD-1521A	Technical Reviews and Audits for Systems, Equipment and Computer Programs
MIL-STD-1530	Aircraft Structural Integrity Program, Airplane Requirements
MIL-STD-1553B	Aircraft Internal Time Division Multiplex Data Bus
MS15002	Fittings, Lubrication (Hydraulic) Surface Check, Straight Threads, Steel, Type II
MS15981	Fasteners, Externally Threaded, Self-Locking, Design and Usage Limitations for

Military (Continued)

MS24665	Pin, Cotter
MS33540	Safety Wiring and Cotter Pinning, General Practices for
MS 33572	Instrument, Pilot, Flight, Basic, Standard Agreement for Helicopters
MS33588	Nuts, Self-Locking, Aircraft Design and Usage Limitations of
MS33602	Bolt, Self Retaining, Aircraft Reliability Maintainability Design and Usage, Requirements for
MS33736	Turnbuckle Assemblies, Clip Locking of

Reports

AFWAL-TR-82-3081 Volume II	Proposed MIL Handbook - Flying Qualities of Air Vehicles
AFWAL-TR-82-3081 Volume I	Proposed MIL Standard - Flying Qualities of Air Vehicles
FAA-RD-76-195	Heffley, Robert K., Stapleford, Robert L., and Rumold, Robert C., "Airworthiness Criteria Development for Powered-Lift Aircraft", MIL-STD-1553 Multiplex Applications Handbook, Air Force Systems Command, Aeronautical Systems Division, Revised 2-1-82

Publications

Military Handbooks

MIL-HDBK-5	Metallic Materials and Elements for Aerospace Vehicle Structures
MIL-HDBK-17	Plastics for Flight Vehicles

Air Force Systems Command Design Handbooks

AFSC DH 1-2	General Design Factors
AFSC DH 1-4	Electromagnetic Compatibility
AFSC DH 1-5	Environmental Engineering
AFSC DH 1-6	System Safety
AFSC DH 2-1	Airframe
AFSC DH 2-2	Crew Stations and Passenger Accommodations

Air Force Regulations Document

AFR-800-14

Vol. I: Management of Computer
Resources Systems

Vol. II: Acquisition and Support Procedures
for Computer Resources in Systems

U. S. Nuclear Regulatory Commission

NUREG-0492, January 1981

Fault Tree Handbook

OTHER PUBLICATIONS

The following documents form a part of this document to the extent specified herein. Unless otherwise indicated; the issue in effect shall apply.

ARINC Characteristics

Part 1 400 Series

429-6

Digital Information Transfer System (DITS)

453

Very High Speed Bus - Draft 2

Part 2 500 Series

581

Mark 1 Air Transport Area Navigation System
June 30, 1970

583

Mark 13 Area Navigation System, Sept. 23, 1976

Part 3 600 Series

600-3

Air Transport Avionic Equipment Interfaces,
April 15, 1981

Part 4 700 Series

701

Flight Control Computer System, March 1, 1979

702-1

Flight Management Computer, January 29, 1980

703

Thrust Control Computer, March 1, 1979

704-3

Inertial Reference System, August 19, 1981

705-3

Attitude and Heading Reference System,
April 22, 1981

706-2	Subsonic Air Data System, February 27, 1981
707-3	Radio Altimeter, March 12, 1981
708-2	Airborne Weather Radar, February 10, 1981
709-4	Airborne Distance Measuring Equipment (DME), April 3, 1981
710-3	Airborne ILS Receiver, August 18, 1980
711-4	Airborne VOR Receiver, April 10, 1981
712-3	Airborne ADF Receiver, April 15, 1981
716-3	Airborne VHF Communication Transceiver, September 25, 1981
717-3	Aircraft Integrated Data System, March 27, 1981
718-3	ATC Transponder (ATCRBS/DABS), September 10, 1981
720-1	Digital Frequency/Function Selection (DFS), July 1, 1980
723-1	Ground Proximity Warning System, August 15, 1981
725-1	Electronic Flight Instruments (EFI), September 5, 1980
726-1	Flight Warning Computer, September 10, 1981
727	Airborne MLS Receiver, Part 1, Aircraft Installation Procedures, November 16, 1979
729-1	Analog and Discrete Data Converter System, September 10, 1981
730-3	Airborne Separation Assurance System, April 3, 1981

National Aircraft Standard

NAS 516	Fitting, Lubrication - 1/8 Inch Drive, Flush Type
---------	--

(Copies of National Aircraft Standards may be obtained from the Aircraft Industries Association of America, Inc., Shoreham Building, Washington, D.C.)

SAE Aerospace Recommended Practices

ARP 926A	Fault Failure Analysis Procedure, SAE Aerospace Recommended Practices
Society of Automotive Engineers, Inc., November 1979	
ARP 926B	Fault/Failure Analysis Procedure for Digital Hardware/Systems, Draft, Jan. 1983
ARP 988	Electrohydraulic Mechanical Feedback Servoactuators
ARP 1281	Servoactuators: Aircraft Flight Controls, Power Operated, Hydraulic, General Specification for
AE4L-81-2, SAE AE4L Committee Report, 15 December 1981	Test Waveforms and Techniques for Assessing the Effects of Lightning- Induced Transients

(Application for copies should be addressed to the American Society of Automotive Engineers, Two Pennsylvania Plaza, New York, New York 10001.)

ICAO Practices

ICAO Annex 10	International Civil Aviation Organization Publication - Aeronautical Telecommuni- cations. Vol. II, Communication Procedures, International Standards, Recommended Practices and Procedures for Air Navigation Services.
---------------	---

RTCA

RTCA Document No. DO-160A Radio Technical Commission for Aeronautics, January 1980	"Environmental Conditions and Test Procedures for Airborne Equipment"
RTCA Document No. DO-178 Radio Technical Commission for Aeronautics, November 1981	"Software Considerations in Airborne Systems and Equipment Certification"

USAF

AFFDL-TR-74-116, January 1975 The Boeing Company, Wichita Division, Wichita, KS 67210	"Background Information and User Guide for MIL-F-9490D Flight Control Systems - Design, Installation and Test of Piloted Aircraft, General Specification for"
---	--

AFFDL-TR-74-116 Sup. 1, January
1980, Northrop Corporation,
Aircraft Group, Hawthorne, CA 90250

"Appendix to Background Information and
User Guide for MIL-F-9490D AFFDL-TR-
74-116"

RADC-TR-82-179, June 1982
Boeing Aerospace Co., Houston, TX

"Sneak Analysis Application Guidelines"

CAA

Subsection D1, British Civil
Airworthiness Requirements,
Civil Aviation Authority,
December 16, 1981

"The Safety Assessment for Systems"

CAA Airworthiness Information
Leaflet, April 25, 1978

"The Certification of Safety Critical
Digital Systems"

British Civil Airworthiness
Requirements Paper No. 367
Issue 3, June 1970

"Airworthiness Requirements for Automatic
Landing Including Automatic Landing in
Restricted Visibility Down to Category 3"

SECTION THREE
SYSTEM LIFE CYCLE

TABLE OF CONTENTS

	Page
SECTION 3	3-1
3. SYSTEM LIFE CYCLE (OVERVIEW)	3-1
3.1 Operational Flight Program (OFF) Development	3-3
3.1.1 Conceptual	3-3
3.1.2 Requirements Definition	3-3
3.1.3 Design and Specification	3-6
3.1.4 Coding	3-8
3.1.5 Testing	3-8
3.1.6 Verification, Validation, Certification, and Qualification	3-13
3.2 Function Criticality	3-17
3.2.1 Criticality Assessment	3-17
3.3 Potential Problem Areas	3-17
3.4 References	3-20

LIST OF ILLUSTRATIONS

Figure		Page
3-1	System Life Cycle	3-1
3-2	Software Activities/Products Relation to System Life Cycle Phases	3-2
3-3	Avionics Software Engineering Process	3-4
3-4	Test Management Milestones	3-7
3-5	Software Activities/Products Relation to Verification/Validation Activities	3-15
3-6	Verification and Validation Activities by Organization Over Software Life Cycle (Ref. 22)	3-16

LIST OF TABLES

Table		Page
3-1	Function Criticality Categories	3-18
3-2	Minimum Verification and Validation Activities Grouped by Criticality (Ref. 24)	3-19

SECTION 3

3. SYSTEM LIFE CYCLE (OVERVIEW)

The phases of the life cycle of a digital system have been given many representations (references 1-3). The basic elements are the same for most representations; the differences are primarily in the terminology. Figure 3-1 presents the approximate time relationship of the life cycle phases considered in this handbook.

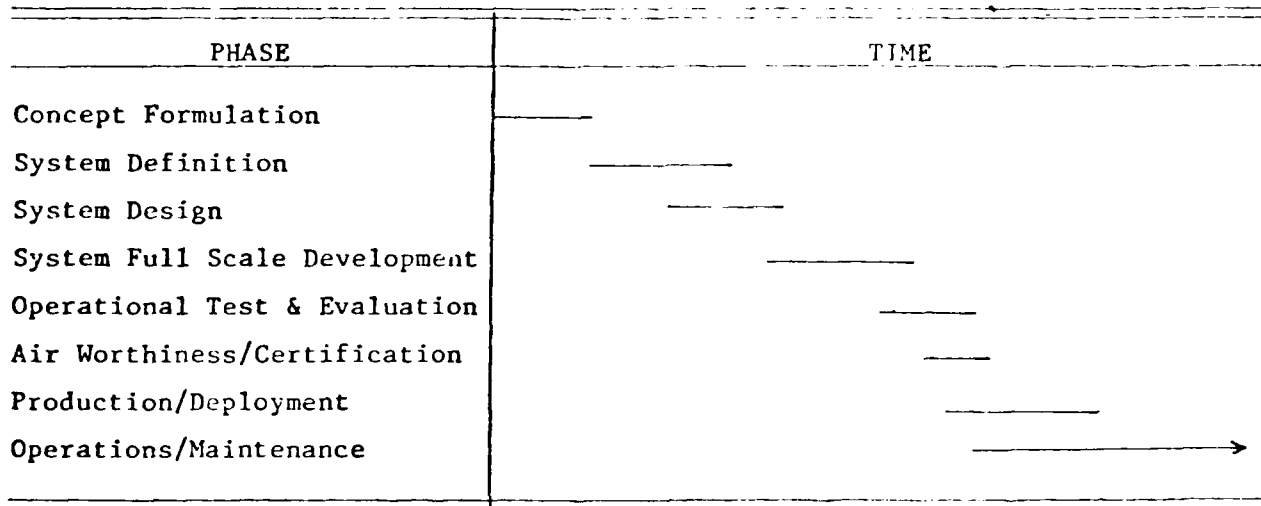


FIGURE 3-1. SYSTEM LIFE CYCLE

During the system life cycle, both the hardware and software elements of the system are designed, developed, used, and maintained. While a digital system is neither hardware nor software alone, many authorities (references 2-12) have chosen to develop a separate representation for the software life cycle. This approach helps in the understanding of the software engineering process, but the user must remember that this handbook is concerned with the complete digital system. Figure 3-2 depicts the relationships between the overall system life cycle phases and the software activities and products during these phases. The primary phases of the software life cycle are:

- | | |
|---------------|------------------------------|
| 1) Conceptual | 4) Code, Test, and Integrate |
| 2) Definition | 5) Evaluation |
| 3) Design | 6) Operations/Maintenance |

Other definitions than the above have been used for the phases by DOD and contractors (reference 2), but most agencies and contractors agree on the activities necessary to develop and maintain highly reliable real-time software.

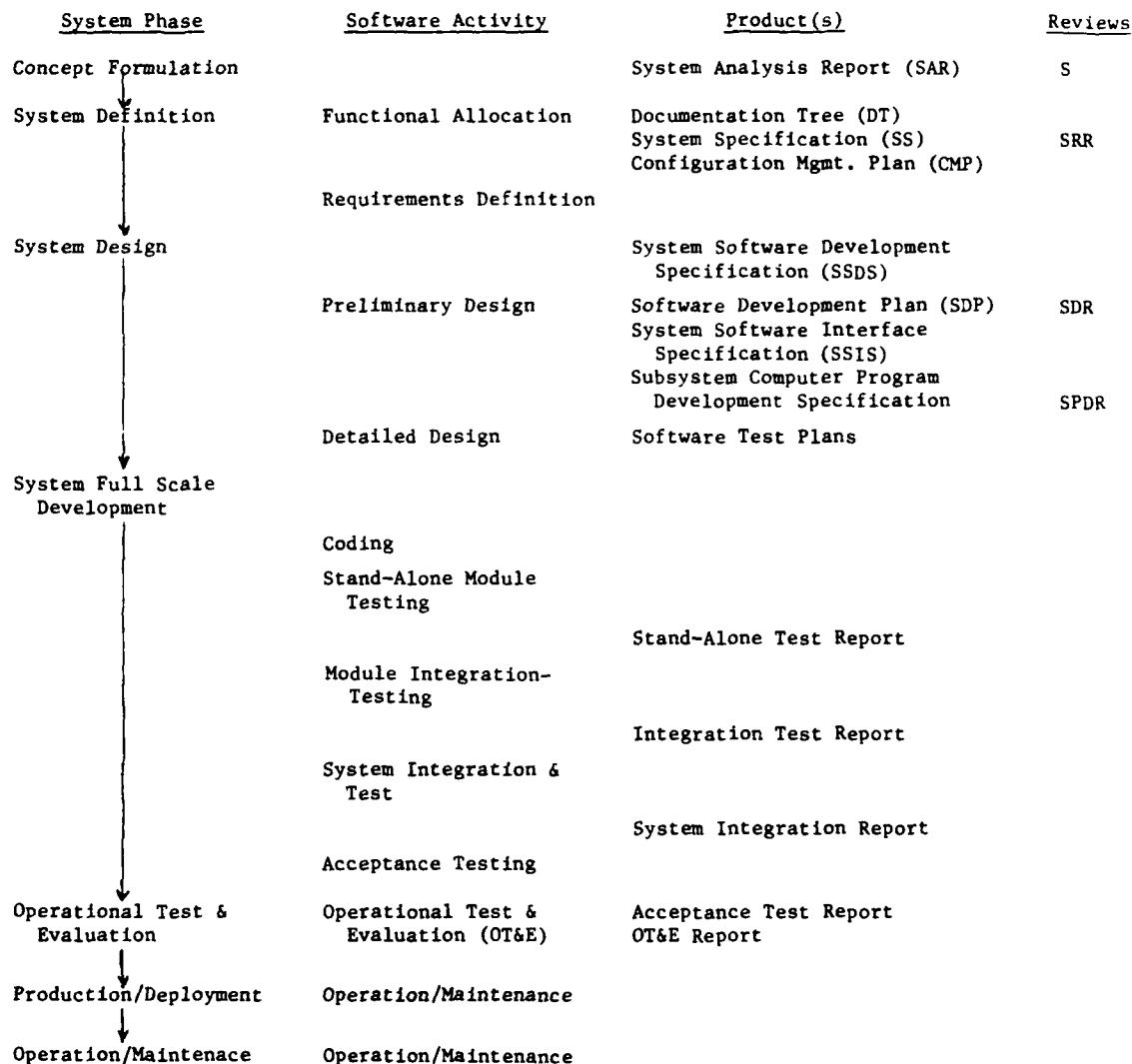


FIGURE 3-2. SOFTWARE ACTIVITIES/PRODUCTS
RELATION TO SYSTEM LIFE CYCLE
PHASES

3.1 OPERATIONAL FLIGHT PROGRAM (OFP) DEVELOPMENT.

The operational flight programs are the programs that reside in the computer or computers used in the digital flight control and avionics systems. As discussed in a subsequent section on system architecture, many of the avionics and flight control systems being developed are federated systems consisting of several computers, each dedicated to a particular function. Fault tolerance may be provided with the federated architecture wherein a computer can perform backup functions in addition to its normal functions in the event of a malfunction of another computer. This federated architecture permits partitioning of the overall system functions to specific computers and the development of operational flight programs for each computer.

The operational flight program development cycle can occur many times throughout the system life cycle. As additional functions or capabilities are required, such as the addition of an improved performance management function, this necessitates modifying the operational flight program in some of the processors. This is necessary since the real-time system involves communication exchanges initiated or driven by the data required by the performance management system.

The major activities in OFP development cycle (references 2-10) include management, functional allocation, support software development, requirements definition, design specification, coding, and evaluation including stand-alone, integration, and system testing followed by flight test. Figure 3-3 depicts the avionics software engineering process which occurs during the OFP development. The right hand column lists many of the major end products of the activities of the OFP development. Major steps in the OFP development are described in the following paragraphs.

3.1.1 Conceptual.

The major technical activities which take place during this phase that precedes the requirements definition are operational analysis and support software development. Tradeoffs are performed between the allocation of the functions to hardware or to software, and between the tasks allocated to specific processors. Functional, performance, interface, and design parameters are assigned to the software component of the system. A simulation of the system functions may be developed to permit a study of the system's ability to satisfy the missions requirements and to support systems tradeoff studies. Support software, such as compilers or system test tools, may also be developed during this time.

3.1.2 Requirements Definition.

Tasks or activities associated with the requirements definition phase of the OFP development depend, to a degree, upon the information available at the initiation of the effort. For a program in which a complete new avionics system and software are to be developed, the requirements definition phase can be quite extensive and require not only a large amount of calendar time but a significant number of man-months and funds. A formal requirements engineering methodology for determining real-time processing requirements (reference 11) may be advantageous. As reported in reference 11, "in almost every software project which fails, the requirements are accused of being late, incomplete, overconstraining, and just plain wrong." In the formal approach to requirements definition taken in reference 11, the

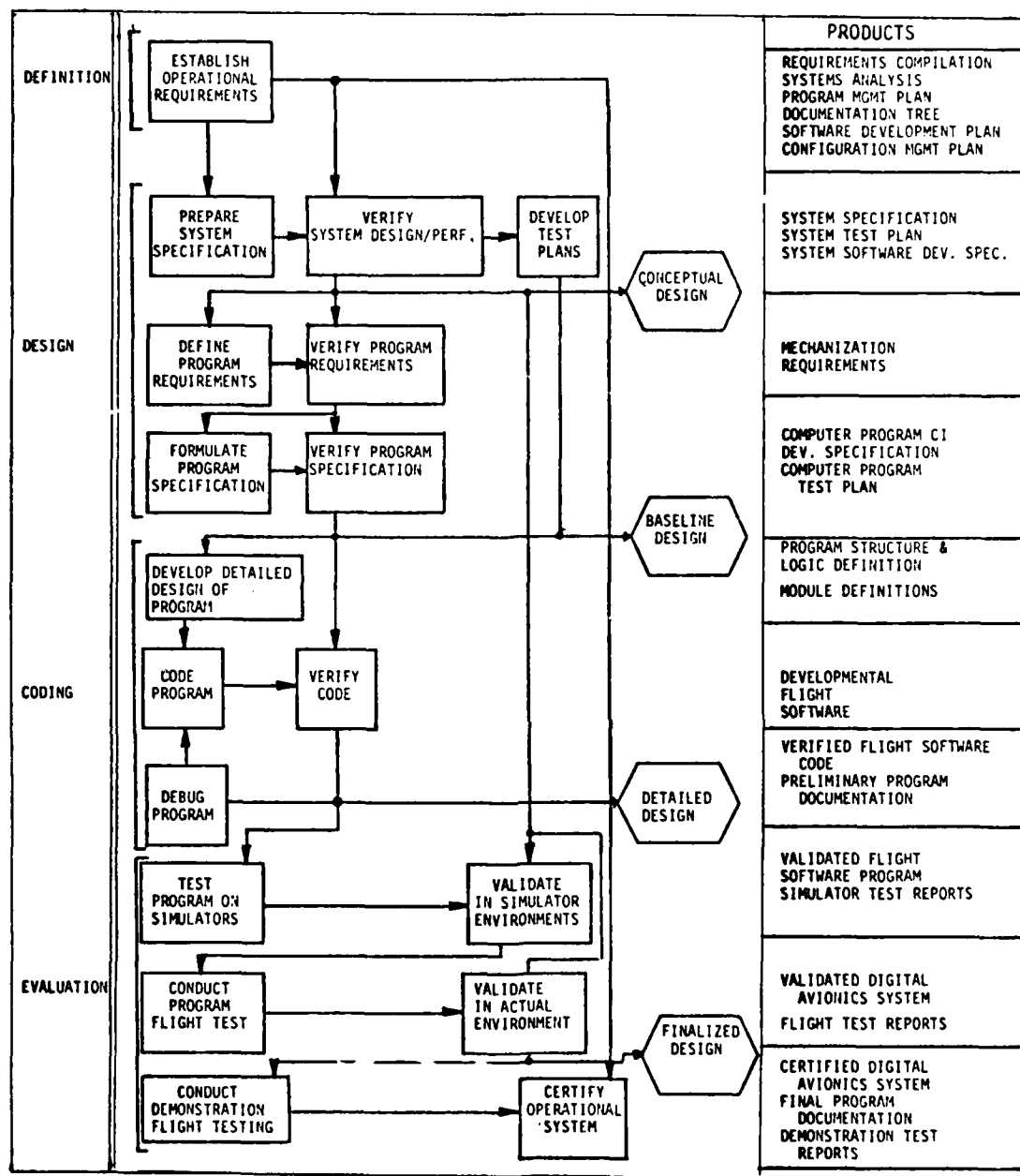


FIGURE 3-3. AVIONICS SOFTWARE ENGINEERING PROCESS

conventional way of describing software requirements, in terms of a hierarchy of its functions, is replaced by a more structured method involving the following key concepts:

(1) Real-time software is tested by inputting an interface message and extracting results of its processing.

(2) Processes to be performed are defined in terms of their relationships of input messages, output messages, processing steps, and data utilized and produced.

(3) A test is defined in terms of the variables measured in the sequence of processing steps connecting the arrival of a message at an input interface to determination of the processing of the message.

(4) Specified sequences of processing (defined as PATHS) can be integrated into a network called a requirements network.

(5) A formal language is used for the statement of requirements based on the above concepts.

(6) Automated tools are used to speed up and validate the requirements.

(7) The methodology's tests produce intermediate results which are evaluated for completeness.

The application of this formal methodology requires (1) establishment of the operational system requirements, (2) development, and transformation of these requirements into functional and performance requirements upon the computer subsystem (including system design, subsystem definition, interface specification, performance allocation to the computer subsystem and identification of system operating and control procedures) and (3) the documentation of these results in a formal requirements document. The requirements contained in this document are then translated and interpreted into a requirements baseline, written in a requirements statement language. Constraints associated with interfaces, timing, etc., are more completely defined and compared with the original requirements document. The requirements are allocated to each of the processing paths and the performance evaluated using a functional simulator of the process. Specific algorithms can be evaluated in the simulator with realistic input data produced by a simulation of the environment.

There are other formal approaches to the requirements definition (references 7-15). Regardless of which of these approaches is elected, the need for a thorough requirements definition must be recognized. If a thorough definition is not provided, the designers will make the missing assumptions and decisions (because they must), in order to accomplish the job.

Meetings with the user should be held to compile any additional requirements not contained in the formal requirements document. An analysis of the system should be conducted to determine the functional architecture of the system. This may be described in terms of interface control documents describing all interfaces between the software modules comprising the functions as well as by the various graphical techniques (reference 16) used to describe the processing requirements. In addition, the software development plan (reference 11) is an output of the requirements definition.

The requirements for documentation should also be established, including not only software documentation but also software test plans and the expected software test reports. At this stage in the software life cycle, a program management plan and configuration management plan should also be available.

Figure 3-3 depicted a flow diagram for the avionics software engineering process. Nearly all software development and maintenance projects adhere to a similar sequence of definition, design, coding, and evaluation as follows:

1. The development group or contractor is normally responsible for:
 - a. The definition of the program requirements.
 - b. Design formulation.
 - c. Coding and checkout.
 - d. Qualification to assure that the program performs the intended functions as stated in the specifications.
2. An independent test group performs in parallel the functions depicted in figure 3-4 including:
 - a. Planning.
 - b. Test requirements definition.
 - c. Tool definition and development.
 - d. Test plan/procedures definition.
 - e. Testing and analysis including program concept analysis, static code analysis.
 - f. Code execution testing and test report preparation.

The objectives of the performance evaluation and assurance analysis testing are interdependent to some extent. The test methodology established during the test requirements definition must be expanded when the test plans are defined and the detailed test procedure is developed.

The test support software tools which are required to accomplish the test objectives are identified during the tool definition and development phase. Automated test techniques and analysis methods must be traded off against the cost in time and money of manual test techniques and analysis methods. After selecting the test support software tools, a method must be established for qualifying them and configuration control procedures selected for the test tools.

The test plans/procedures definition requires a detailed analysis of how each test will be performed. The test plan should include:

- (1) Test support software or hardware test bed to be used.
- (2) Test scenarios.
- (3) Test criteria for both performance and quality assurance.
- (4) Detailed test procedures.
- (5) Control of data standards and software deliverables.

3.1.3 Design and Specification.

Figure 3-3 also depicted the many tasks which must be performed during the design and specification phase. The requirements previously defined are input to the

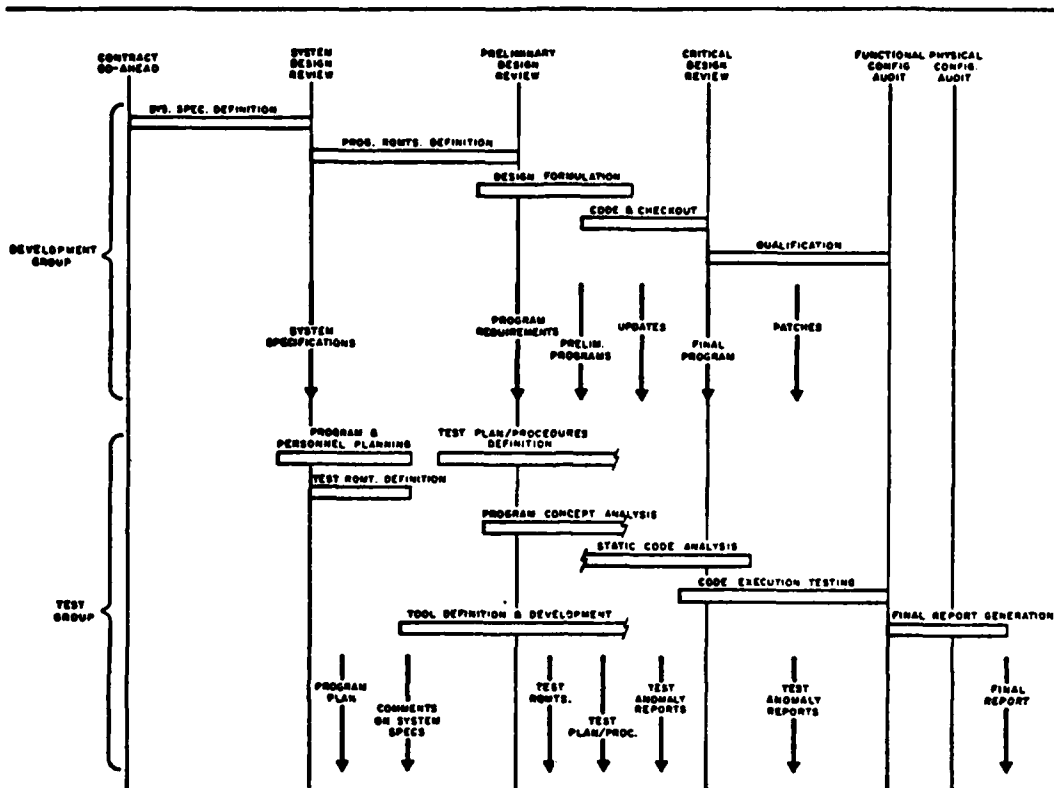


FIGURE 3-4. TEST MANAGEMENT MILESTONES

system design process. This process entails analysis, synthesis of candidate configurations and evaluation of these candidate configurations to arrive at a system design which meets the requirements. These requirements should be fully expressed in the systems specification which should be prepared in accordance with MIL-STD-490 or the format in appendix A. After verification of the system design to assure that it meets the performance requirements, the operational flight program (OFP) requirements should be defined. These requirements should then be verified prior to formulation of the programs specification. The program specification is then verified to assure that the program requirements are met. In addition, the system and OFP test plans should be developed during this phase. Primary outputs of this phase are the system specification, system test plan, system software development specification (part 1), OFP mechanization requirements, OFP Critical Item (CI) (Subsystem Computer Program) development specifications, and OFP test plan. At the completion of the design and specification phase, the baseline design should have undergone review and approval.

3.1.4 Coding.

Prior to the beginning of the actual coding in whatever language is selected, final detailed design of the program should be completed. This includes completion of the program structure and logic definition as well as definition of each of the software modules. Any effort devoted to actual source code generation prior to completion of all of the foregoing tasks should be held to a minimum. The common mistake in the coding phase is to begin coding prior to a complete development of all specifications and requirements. Changes in requirements often result in code being thrown away or in an extensive effort to salvage the existing code and include the revisions necessary to satisfy the revised requirement.

After coding of the program, the code is verified to ascertain if the program specifications are met. With normal development, the program will require debugging and code changes in the verification. These tests often overlap the next phase in the OFP development cycle.

3.1.5 Testing.

The planning phase includes planning for staffing and preparation of schedules including those for:

- (1) Major project design and technical reviews.
- (2) Dates and contents of data package deliveries.
- (3) Dates of deliverable data items and other major milestones.
- (4) Dates for completion of important test support software tools including definition dates for the test tool requirements, design flows, description document, and users manual.
- (5) Schedule date for review of activities.

The man loading for each of the support activities is prepared after the schedule and work breakdown structure are prepared.

The test requirements definition involves clearly identifying program requirements whose violation could compromise system safety or security. Two types of testing

are generally performed. Performance evaluation testing is performed to measure the efficiency (timing and memory), maintainability (ease of modification), accuracy (numerical and logical), compatibility with system and user (convenience, vulnerability), and testability. Performance evaluation involves measuring the extent to which the software:

- (1) Satisfies system requirements.
- (2) Satisfies program end item requirements.
- (3) Is designed and coded efficiently.
- (4) Degrades the performance of the system or the subsystem in which it operates.

In order to conduct a performance evaluation, it is very important that the system and end item requirements be fully defined.

The other type of testing is an assurance analysis conducted to ascertain that the program performs all of its intended functions and does not perform unintended functions that could degrade or compromise the safety or the security of the system to which it belongs. Assurance analysis involves the determination of the extent to which the software contains coding which could contribute to:

- (1) Failure to respond in a timely fashion to critical program conditions.
- (2) Operations that present a hazard to equipment or personnel.

The objectives of the performance evaluation and assurance analysis testing are interdependent to some extent. The test methodology established during the test requirements definition must be expanded when the test plans are defined and the detailed test procedure is developed.

The test support software tools which are required to accomplish the test objectives are identified during the tool definition and development phase. Automated test techniques and analysis methods must be traded off against the cost in time and money of manual test techniques and analysis methods. After selecting the test support software tools, a method must be established for qualifying them and configuration control procedures selected for the test tools.

The test plans/procedures definition requires a detailed analysis of how each test will be performed. The test plan should include:

- (1) Test support software or hardware test bed to be used.
- (2) Test scenarios.
- (3) Test criteria for both performance and quality assurance.
- (4) Detailed test procedures.
- (5) Control of data standards and software deliverables.

3.1.5.1 Stand-Alone Testing of Modules. During the testing and analysis phase, a review of all discrepancies issued by the test group should be conducted and each discrepancy evaluated for correctness. If a change is required in the program code or support tools or the program requirements change, a decision must be made whether it will be necessary to repeat tests and analyses already completed. Careful configuration control must be maintained in order to properly conduct the the testing and analysis, static code analysis, and code execution testing. The

purpose of the program concept analysis is to ascertain that the program has been adequately and accurately designed. This involves specifying the standards to be followed in designing and writing the program including number systems, specification language, flow charting standards, programing standards, and program change procedures. The methods generally used to perform concept analysis testing include:

(1) Documentation research and analysis which consists of verifying that the program requirements are genuine and related to the real system requirements, that variables and parameters are consistent across all documents, that all data can be traced to consistent, coherent sources.

(2) Algorithm analysis which consists of verifying that the design will work, is accurate, and truly reflects the program requirements.

Discrete and continuous simulations are often used to check out concepts and algorithms, and independent sizing and timing analyses should be performed for real-time programs.

Upon completion of the coding and debugging of every software module, stand-alone tests should be conducted in accordance with the OFP test plan. These tests may involve static analysis, dynamic testing, symbolic execution, and proofs of correctness (reference 17). In systematic testing effort, the specifications are the standard against which the program is validated. These specifications include the requirements specification and the design specifications.

3.1.5.1.1 Static Analysis. The static analysis is normally conducted prior to dynamic testing and reduces the number of dynamic test cases required. Information obtained by static analysis (reference 17) includes:

- "(1) Syntactic error messages.
- (2) Number of occurrences of source statements by type.
- (3) Cross-reference maps of identifier usage.
- (4) Analysis of how the identifiers are used in each statement (data source, data sink, calling parameter, dummy parameter, subscript, etc.).
- (5) Subroutines and functions called by each routine.
- (6) Uninitialized variables.
- (7) Variables set but not used.
- (8) Isolated code segments that cannot be executed under any set of input data.
- (9) Departures from coding standards (both language standards and local practice standards).

(10) Misuses of global variables, common variables, and parameter lists (incorrect number parameters, mismatched types, uninitialized input parameters, output parameters not assigned to, output parameter assigned to but never used, parameters never used for either input or output, etc.)."

Static code analysis can be done manually or by automated methods. Static code tools include comparators which compare the code of one program version to that of another and reveal any differences; editors which analyze source code for coding errors including setting and clearing flags, use of error-prone instruction sequences, proper calling sequences, attempts to reference and modify restricted data, use of restricted instructions, or use of inaccessible instructions. Flow charts can be used to reconstruct the logic flow of both higher-level source language programs and assembly language programs. Logic/equation generators automatically reconstruct arithmetic and flow chart assembly language programs. Program structure analyzers are used to analyze the program paths under input control. Symbolic program executors decompose source code by logically executing the code.

The limitations of static analysis include the inability to evaluate subscripts or pointers, and the inability to identify all semantic paths which are subsets to the syntactic paths.

3.1.5.1.2 Dynamic Analysis. Dynamic testing is normally used to validate program functions, since the exact sequence of values assigned to variables can be recorded while the program is operating in real-time. The following types of tests comprise dynamic testing:

(1) Functional testing conducted to demonstrate that the software performs satisfactorily under normal operating conditions by computing nominally correct output values given nominal input values.

(2) Logical tests in which arithmetic, error handling, initialization, interfaces, and timing are tested.

There are three strategies for the dynamic testing. These are bottom-up, top-down, and mixed testing. Each of these strategies has application across the stand-alone, integration, and system testing phases.

Code execution testing may be done on a host computer which simulates or emulates the target computer; or the actual execution may be done on the target machine. This testing is normally performed either bottom-up or top-down. Bottom-up testing involves testing the subprogram modules and then successfully linking these modules together until the total program has been tested. The most common tools used to test software at the module level are test drivers, simulations, and execution instrumenters. These instrumenters may be hardware monitors or software "probes" within the source code.

The sequence of testing individual modules proceeds through tests such as the following:

(1) Initialization tests in which the performance of all initialization operations are tested.

(2) Interaction tests in which all quantities, variables, and systems conditions obtained from other modules are examined to determine the sensitivity of the module under test to the possible values or states.

(3) Arithmetic tests in which precision of arithmetic calculations are checked to discover where insufficient precision is maintained or incorrect arithmetic calculations are performed.

(4) Timing analyses in which the longest and shortest possible execution time for all tests are determined to establish the execution time requirements for the module to identify potential timing problems.

(5) Branch logic tests in which the correct branching decision paths for each branch and each closed-looped test case are checked.

In real-time systems such as avionics, the most critical problems may be in software elements and modules at the bottom of the structural hierarchy (reference 18). A bottom-up testing approach is advocated if these time-critical conditions are present. After these modules are tested, the testing can return, if desired, to the top-down philosophy.

Top-down testing, used in the stand-alone test phase begins with the main program that requires the use of dummy modules (program stubs) to simulate the effect of routines below the level of those being tested. This is particularly appropriate to communication between processors in a federated system since the data exchanges and interrupt handling will exercise a control or executive program.

Mixed testing may be used to save test time. Significant amounts of machine time can often be saved by conducting stand-alone tests on a module before inserting it into the top-down structure. In some cases, where top-down testing may be used, it is necessary to test certain low-level modules first. While mixed testing is predominately top-down, the use of the bottom-up techniques on certain modules and subsystems alleviates many of the problems of pure top-down testing.

3.1.5.2 Integration Testing. Once the modules have completed individual code execution testing, they are integrated into the complete software component package and tested as a group. At this level, the testing is primarily functional. The testing often takes place in the laboratory containing the target computers and enough equipment to simulate the application with considerable fidelity.

Integration testing is normally conducted using dynamic test methods. If the bottom-up testing strategy is used, the modules that were previously tested in the stand-alone phase are integrated one at a time to verify the operation of the interfaces between modules including data, control, and service interfaces. The lower level modules are successively combined to form higher level routines. If a bottom-up testing strategy is used, the test plan must clearly delineate the sequence of integration tests and the driver software required for testing at each step. A significant amount of test software is required for the bottom-up approach.

Top-down dynamic testing is applicable if a top-down structured system design has been followed. Testing is then distributed throughout the system development

cycle. The top level interfaces are tested first and most often with dummy modules used for the routines below the level of those being tested. The top level routine provides a natural test harness for the lower level routines and the errors can be isolated to the new modules and interfaces being integrated.

As each test is conducted, a test report will be generated. After all testing is completed, a final report is generally generated which includes all errors detected and the status of their correction.

3.1.5.3 System Integration and Testing. The system integration and testing phase consists of testing the completed OFP in the flight computer in a dynamic simulation environment. These system tests shall be conducted in accordance with the system test plan and precede the acceptance testing and engineering flight test phase. These tests may also be conducted in parallel with an Independent Verification and Validation (IV&V) effort.

3.1.5.4 Engineering Flight Test. After completing the ground-based testing of the OFP operating in the actual flight computer with the other simulated or real subsystems, an engineering flight test program will be conducted in accordance with the flight test plan. Should the flight test reveal the need for changes in the OFP, the change would normally be made to the appropriate module or modules and the sequence of stand-alone, integration, and system tests repeated prior to flight testing of the change. Upon completion of the engineering flight test program, the OFP development cycle is considered to be complete.

3.1.6 Verification, Validation, Certification, and Qualification.

Verification and validation are critically important to real-time systems such as avionics and flight control systems. Verification is concerned with demonstrating the logical correctness and showing that the program performs according to its specifications. Validation is a process of demonstrating through testing in the real environment (or an environment nearly as real as possible) that the system not only is verifiable but also satisfies the user's requirements. In this sense, validation encompasses verification (reference 19). As stated in reference 19, there are a number of definitions of verification and validation. There seems to be no complete consensus on these definitions. Section 7 of the handbook presents information on different definitions of verification and validation.

The term "certification" similarly requires definition. Reference 20 defines certification as "connotating an authoritative endorsement implying written testimony that the program is of a certain standard or quality." This definition is consistent with the foregoing definitions of verification and validation. Qualification entails ensuring that a product meets or exceeds a minimum industrial and/or commercially accepted level of excellence (reference 21). This presupposes that the standards against which the quality can be measured are in existence or can be developed.

The overlap between verification, validation, certification, and qualification is one of degree, since the activities take place in a sequential manner throughout the life cycle of the software and do not necessarily involve the same organization or group performing these activities. The following section discusses the activities of verification, validation, certification, and qualification.

3.1.6.1 Activities (What is Done/When It Is Done). Verification activities take place at the initiation of the software life cycle and continue throughout the life cycle. Validation activities encompass these verification activities and occur later in the software life cycle after the integration testing begins. Certification activities take place after the software has been verified and been validated. Qualification activities nearly always follow the verification and validation activities as well as the certification and recertification as shown in figure 3-5. While the figure implies a sequential set of software activities and verification and validation activities with no overlap, in reality, there is some overlap between the completion of one activity and the initiation of the next activity as previously shown in figures 3-1 and 3-4.

Note that the block in the verification and validation activities referring to systems certification and recertification are not performed by the group performing the previous verification and validation activities. The verification and validation activities are often performed by an Independent Test Organization (ITO) which may be an in-house group separate from the software development group that reports directly to the project manager. In some cases, the ITO is another contractor or government agency. The system certification/recertification is the responsibility of the regulatory agency. The system qualification activity is normally the responsibility of the user or purchaser of the equipment.

Verification and validation involves activities over the entire software life cycle and not just during the testing. References 21 and 22 recommend that the verification and validation be performed by an independent organization.

As depicted in figure 3-5, the test and evaluation is performed after completion of acceptance testing and system validation testing by the independent test organization. Reference 21 states "test and evaluation is defined here as discipline imposed outside the 'doing' project organization to independently assess whether a product fulfills/meets objectives by executing test plans and/or procedures, specifically in support of the end user; it entails evaluating a product's performance in at least a nearly live environment. Some organizations formally turn over test and evaluation responsibility to an organization outside the 'doing' project organization after the product reaches a certain stage in development, their philosophy being that developers cannot be objective to the point of fully testing/evaluating what they have produced." Reference 23 states "a programmer should avoid attempting to test his or her own program. A programming organization should not test its own program."

Figure 3-5 illustrates the interaction between the groups performing the software activities and the verification and validation activities. Both groups are involved in the verification and validation activities which occur over the software cycle. Figure 3-6 (reproduced from reference 22) depicts some of the interactions in a different manner. This figure also depicts the reviews and audits and the role of the various organizations in the conduct of the reviews and audits. For example, the preliminary design review is conducted by the system engineering organization that originally established the system requirements. The software development group participates in the preliminary design review which is also attended by the ITO.

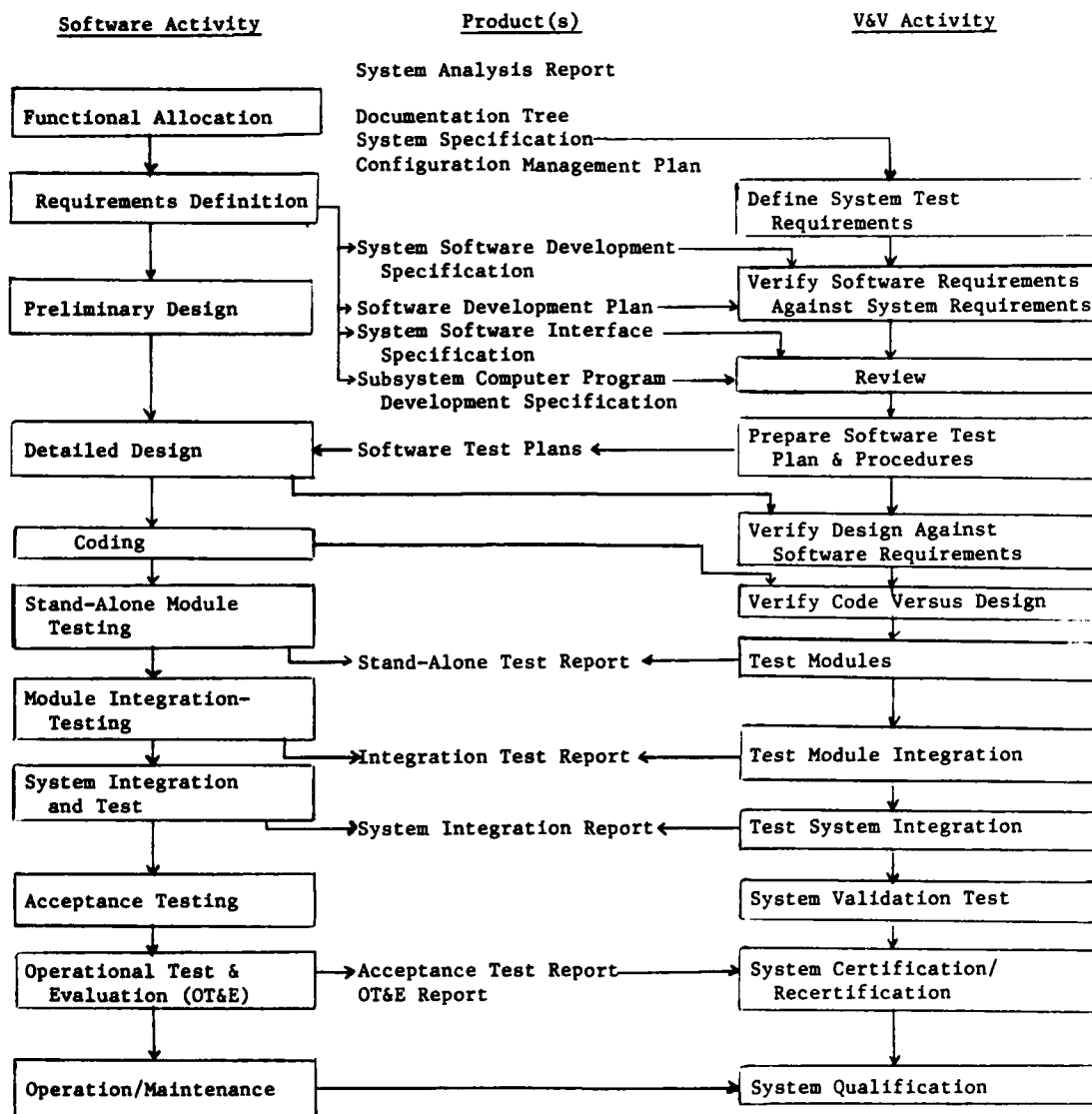


FIGURE 3-5. SOFTWARE ACTIVITIES/PRODUCTS RELATION TO VERIFICATION/VALIDATION ACTIVITIES

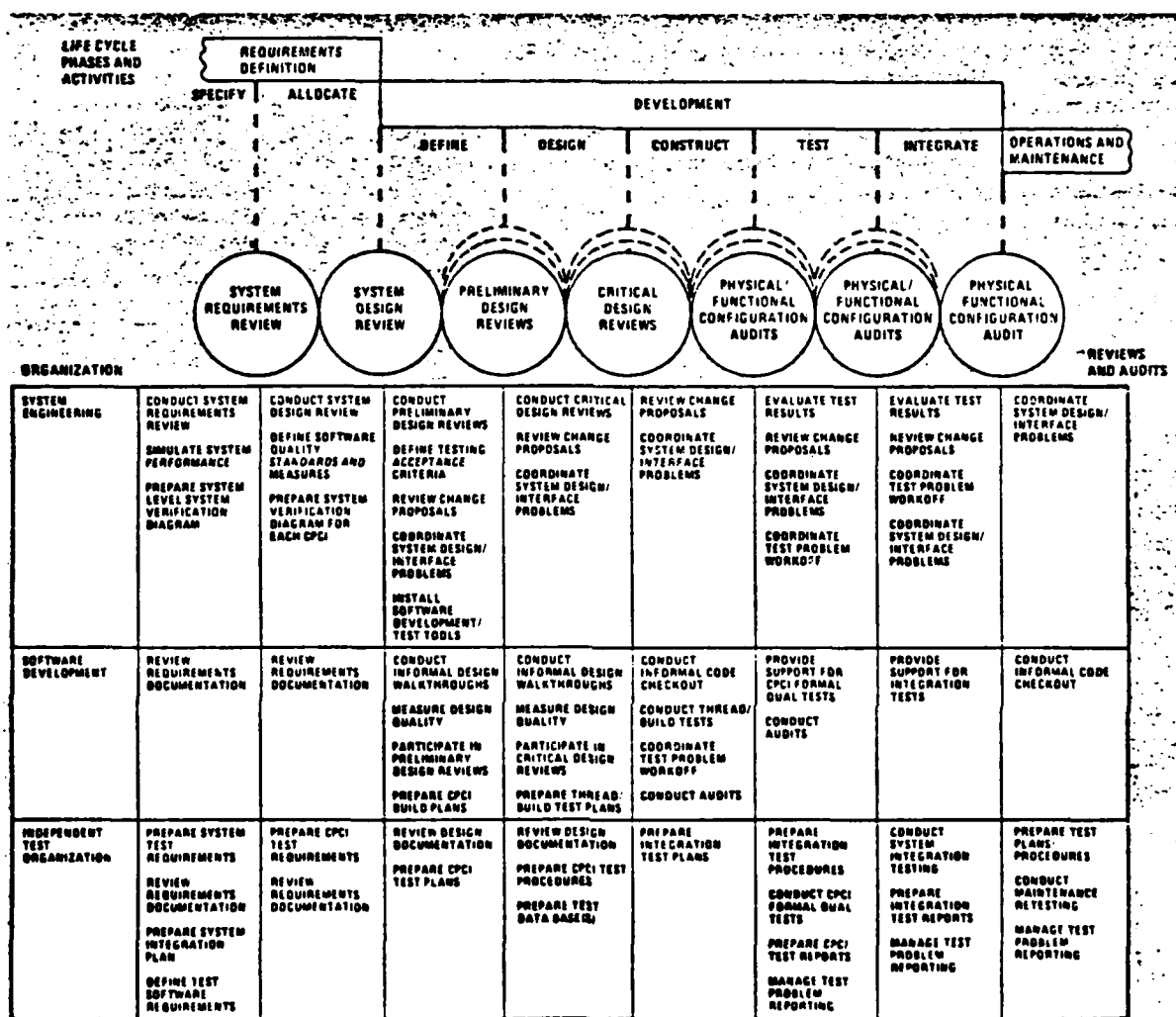


FIGURE 3-6. VERIFICATION AND VALIDATION ACTIVITIES BY ORGANIZATION OVER SOFTWARE LIFE CYCLE (Ref. 22)

3.2 FUNCTION CRITICALITY.

As discussed in Section 7 of this handbook and reference 24, there has been some controversy regarding which software functions require the entire sequence of verification and validation activities. Attempts have been made to classify the avionics and flight control functions in terms of their criticality with some controversy concerning the number of levels of criticality as well as the specific definition associated with each term as illustrated by table 3-1.

Table 3-2 (reproduced from reference 24) depicts a matrix of verification and validation activities versus the function criticality category.

3.2.1 Criticality Assessment.

The determination of which functions are flight critical should be done by the organization establishing the system requirements and producing the system specification. Reference 24 suggests that the system requirements document may originate from either the installer or the equipment manufacturer. This is of particular importance in case the software is modified after delivery. As discussed in Section 7, various groups have suggested that only the flight critical software need be recertified and that essential and nonessential software resident in the same processor with the flight critical software were changed. This presumes that the recommendations in reference 24 for configuration management will be adopted by both the users and regulatory agencies. These recommendations include "each memory component should have program identification in the first few memory locations." Reference 24 also states that "each memory device within a unit should be physically marked such that both hardware and software status may be identified." Designers of fault tolerant systems have suggested that this requires absolute code locations and does not permit relocatable codes. These designers feel that this approach is incompatible with many fault tolerant system designs which are needed to achieve the flight safety requirements of the regulatory agencies. Additional research will be needed to settle the question of the need to recertify all of the software in a specific computer if any of the software is changed or modified.

3.3 POTENTIAL PROBLEM AREAS.

Many end users, such as airlines, have identified their intention to modify the operational flight programs which make up the avionics and flight control software. These software maintenance activities may or may not require recertification of the software, depending upon the final regulatory agency requirements regarding whether essential and nonessential software resident in a computer with critical software will require recertification as long as the critical software supposedly is unchanged. The potential exists that a software modification which outputs data to another module or interfaces with the overall system executive may be considered to impact the flight critical software.

TABLE 3-1. FUNCTION CRITICALITY CATEGORIES

RTCA (Ref. 24)	FAA (Ref. 25)	MIL-F-9490D (Ref. 26)	NASA (Ref. 27,28)
<u>Critical</u> - Functions for which failures or design errors would prevent continued safe flight or landing	<u>Critical</u> - Functions which would prevent the continued safe flight and landing of the airplane if not properly accomplished. Failure conditions which result in improper accomplishment or loss of critical functions must be extremely improbable	<u>Essential</u> - A function is essential if loss of the function results in an unsafe condition or inability to maintain minimum safe operation	<u>Flight Critical</u> - Required for Safe Flight
<u>Essential</u> - Functions for which failures or design errors would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions	<u>Essential</u> - Functions which would reduce the capability of the airplane or the ability of the flight crew to cope with adverse operating conditions if accomplished improperly or lost. Failure conditions which result in improper accomplishment or loss of essential functions must be improbable.	<u>Flight - Phase Essential</u> - A function is flight phase essential if loss of the function results in an unsafe condition or inability to maintain minimum safe operation only during specific flight phases	<u>Flight-Crucial</u> - Required to initiate CAT II or III approach and landing. Loss results in reduced operative performance and increases crew workload. Not required for dispatch unless adverse weather expected during flight.
<u>Non-Essential</u> - Functions for which failures or design errors could not significantly degrade airplane capability or crew ability.	<u>Non-Essential</u> - Functions which could not significantly degrade the capability of the airplane or the ability of the flight crew to cope with adverse operating conditions if accomplished improperly or lost. Failure conditions which result in improper accomplishment or loss of non-essential functions may be probable.	<u>Non-Critical</u> - A function is noncritical if loss of the function does not affect flight safety or result in control capability below that required for minimum safe operation.	<u>Non-Flight Critical</u> - Required for energy efficient and on-time flight. Not required for dispatch.

TABLE 3-2. MINIMUM VERIFICATION AND VALIDATION ACTIVITIES
GROUPED BY CRITICALITY (Ref. 24)

VERIFICATION OR VALIDATION ACTIVITY	CRITICALITY CATEGORY		
	CRITICAL	ESSENTIAL	NON-ESSENTIAL
VERIFY SOFTWARE REQUIREMENTS VERSUS SYSTEM REQUIREMENTS	REQUIRED ⁽¹⁾	(2)	RECOMMENDED ⁽³⁾
VERIFY DESIGN VERSUS SOFTWARE REQUIREMENTS	REQUIRED	(2)	RECOMMENDED
VERIFY CODE VERSUS DESIGN	REQUIRED	(2)	RECOMMENDED
TEST SOFTWARE MODULES	REQUIRED	(2)	RECOMMENDED
TEST MODULE INTEGRATION	REQUIRED	(2)	RECOMMENDED
TEST HARDWARE/SOFTWARE INTEGRATION	REQUIRED	REQUIRED	RECOMMENDED
SYSTEM VALIDATION TESTING ⁽⁴⁾	REQUIRED	REQUIRED	REQUIRED

1. REQUIRED indicates that it is necessary to produce evidence that this activity has been carried out.
2. In these cases, the extent of the verification or validation activities required will be dependent on the severity of the effects of design error. The assessed severity may vary from one regulatory agency to another.
3. RECOMMENDED indicates that although evidence that this activity has been carried out need not be available, good software practice would suggest that it should take place in an orderly fashion.
4. The ISO approval process may not require separate system validation tests.

3.4 REFERENCES.

1. Dickman, Sidney, and Katzenberger, Francis X., "Systems Engineering," Sperry Rand Engineering Review, Vol. 21, Number 1, 1968.
2. "Management Guide to Avionics Software Acquisition, Vol. II--Software Acquisition Process," ASD-TR-76-11, Volume II, Logicon, Inc., June 1976.
3. Jensen, Randall W., and Tonies, Charles C., Software Engineering, Prentice-Hall, Incorporated, Englewood Cliffs, NJ, 1979.
4. Hitt, Ellis F., Bridgman, Michael S., et al, "F-111A/E Digital Bomb-Nav System Software Analysis," AFAL-TR-79-1043, Battelle-Columbus Laboratories, Columbus, OH, January 1979.
5. Patterson, Alt E., and Long, Eugene M., "The F-111 OFP Dynamic Simulation System," Proceedings of the IEEE 1976 National Aerospace and Electronics Conference, Pages 912-919.
6. Patterson, Alton E., and Algire, Richard G., "An Approach to Modern Avionics Integration Support."
7. De Roze, Barry C., and Nyman, Thomas H., "The Software Life Cycle--A Management and Technological Challenge in the Department of Defense," IEEE Trans. on Software Engineering, Vol. SE-4, No. 4, Pages 309-318 (July 1978).
8. Cooper, John D., "Corporate Level Software Management," *ibid*, Pages 319-326.
9. Cave, William C., and Salisbury, Alan B., "Controlling the Software Life Cycle--The Project Management Task," *ibid*, Pages 326-334.
10. McHenry, Robert C., and Walston, Claude E., "Software Life Cycle Management: Weapons Processor Developer," *ibid*, Pages 334-344.
11. Alford, Mack W., "A Requirements Engineering Methodology for Real-Time Processing Requirements," IEEE Transactions on Software Engineering, Vol. SE-3, No. 1 (January 1977).
12. Davis, C. G., and Vick, C. R., "The Software Development System," *ibid*, Pages 69-84.
13. Ross, Douglas T., and Schoman, Kenneth, E., Jr., "Structured Analysis for Requirements Definition," *ibid*, Pages 6-15
14. Hamilton, M., and Zeldin, S., "Higher Order Software--A Methodology for Defining Software," IEEE Trans. Software Eng., Pages 9-32 (March 1976).
15. Tiechroew, D., Hershey, E., and Bastarache, M., "An Introduction to PSL/PSA," ISDOS Working Paper 86, Dept. Industrial and Operations Eng., Univ. of Michigan, Ann Arbor (March 1974).
16. Kodres, Uno R., "Analysis of Real-Time Systems by Data Flowgraphs," IEEE Transactions on Software Engineering, Vol. SE-4, No. 3, Pages 169-178 (May 1978).

17. Fairley, Richard E., "Static Analysis and Dynamic Testing of Computer Software," IEEE Computer, Vol. 11, No. 4, Pages 14-23 (April 1978).
18. Yourdon, Edward, and Constantine, Larry L., Structured Design, Prentice-Hall, Incorporated, Englewood Cliffs, NJ 07632, 1979.
19. Hitt, Ellis F., "Definition of Verification and Validation," Letter to FAA, ACT-340, Battelle-Columbus Laboratories, March 19, 1981.
20. Hetzel, William C., Program Test Methods, Page 9.
21. Bersoff, Edward H., Henderson, Vilas D., and Siegel, Stanely G., Software Configuration Management, An Investment in Product Integrity, Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, 1980.
22. Deutsch, Michael S., "Software Project Verification and Validation," IEEE Computer, April 1981, Pages 54-70.
23. Myers, Glenford J., The Art of Software Testing, John Wiley and Sons, Incorporated, New York, 1979.
24. "Software Considerations in Airborne Systems and Equipment Certification," Document No. RTCA/DO-178, SC 145, Radio Technical Commission for Aeronautics, November, 1981.
25. "Airplane System Design Analysis," Advisory Circular 25.1309-1, Dept. of Transportation, Federal Aviation Administration, September 7, 1982.
26. "Flight Control Systems-Design, Installation, and Test of Piloted Aircraft, General Specification for," MIL-F-9490D (USAF), June 6, 1975.
27. "Validation Methods for Fault-Tolerant Avionics and Control Systems--Working Group Meeting I," NASA Conference Publication 2114, March 12-14, 1979.
28. Hitt, E. F., Bridgman, M. S., and Robinson, A. C., "Comparative Analysis of Techniques for Evaluating the Effectiveness of Aircraft Computing Systems," NASA CR-159358, Battelle-Columbus Laboratories, April 1981.

SECTION FOUR
MISSION FACTORS

TABLE OF CONTENTS

	Page
SECTION 4	4-1
4. MISSION FACTORS	4-1
4.1 Mission Environment	4-1
4.1.1 Airline Operations	4-1
4.1.2 Airport Configuration	4-4
4.1.3 Air Traffic Mix/Density at Specific Airports as a Function of Time-of-Day	4-4
4.1.4 Air Traffic Control/Navigation Aids	4-4
4.1.5 Operating Time Considerations in Validation	4-5
4.2 Atmospheric Environment	4-6
4.2.1 Lightning	4-7
4.3 Electromagnetic Interference	4-8
4.4 Safety Requirements	4-8
4.5 Constraints	4-15
4.5.1 Aircraft	4-15
4.5.2 Airport	4-15
4.5.3 Approach/Landing Performance Limits	4-15
4.5.4 Airline Imposed Constraints	4-17
4.6 Advanced Digital Integrated Flight Control and Avionics System Functions	4-17
4.6.1 Flight Control Background	4-17
4.6.2 Command Generation (Guidance) Background	4-17
4.6.3 State Estimation (Navigation) Background	4-18
4.6.4 Aircraft System Functions	4-19
4.7 References	4-22

LIST OF ILLUSTRATIONS

Figure		Page
4-1	Flux Linkages Versus Conductor Position (Reference 12)	4-9
4-2	Conductor Routing (Reference 12)	4-10
4-3	Frequency Considerations	4-10
4-4	Relationship Between Probability and Severity of Effects (Reference 29)	4-13
4-5	Block Diagram of Navigation, Guidance, and Control Outer Loop	4-20

LIST OF TABLES

Table		Page
4-1	Equipment Required for IFR Dispatch (FAR Parts 25.1303 and 121.303)	4-2
4-2	Airborne Equipment Requirements Categories (Categories I and II ⁺)	4-3
4-3	Primary Radio Navigation Systems	4-5
4-4	Example of Digital Flight Control and Avionics System Operating Profile	4-6
4-5	Protection from Indirect Electromagnetic Effects	4-8
4-6	EMI Sources and Susceptibility of a Hypothetical Digital Avionics Subsystem	4-11
4-7	How to Improve EMC of the Subsystem	4-12
4-8	Quantitative Safety Requirements	4-12

SECTION 4

4. MISSION FACTORS

The mission factors that must be taken into account in the validation of digital integrated flight control and avionics systems include the mission environment, atmospheric environment, safety requirements, constraints, and the advanced digital integrated flight control and avionics systems functions required to satisfy the mission and safety requirements while operating within the constraints. Each of these is discussed in this section.

4.1 MISSION ENVIRONMENT.

The elements of the air transportation system include the air vehicle and its onboard systems and crew, the airports, the air traffic control system which controls the movement of the air vehicle through the airspace, and the ground access. The aspects of these elements which interact with, and hence either are impacted by or impact the validation of, the digital flight control and avionics systems are discussed in succeeding paragraphs.

Safety and reliability analyses are directly based up the operating time of the digital flight control and avionics systems. Hence, those aspects of the air transportation system elements which influence the operating time and the time between inspection should be taken into consideration.

4.1.1 Airline Operations.

Airline operations normally involve, for a single aircraft, operation over a route network with selected cities served by that aircraft at specific scheduled times of day. The route network may consist of a single city pair, with the aircraft shuttling passengers between these cities with a typical day consisting of 10 hours of operation with four round trips per day, giving a total of eight takeoffs and landings each day. The network could involve service to four cities with only two complete trips around the network in a single day. The time required for a single flight between two airports is not only a function of the distance between the airports and the speed of the aircraft, but also a factor of the horizontal and vertical flight trajectory components and delays encountered in the airspace and on the airport surface.

4.1.1.1 Inspection/Maintenance Policy. FAR Parts 25.1303 (reference 1) and 121.303 (reference 2) provide guidance information on the equipment required for IFR dispatch. Table 4-1 lists the subsystems and the number required operable at dispatch when operating under instrument flight rules (IFR).

Table 4-2 was extracted from FAA Advisory Circular 120-20 (reference 3) and lists the airborne equipment requirements for Categories I and II operations. Category III requirements are given in AC 120-28C (reference 4). Automatic landing system criteria are given in AC 20-57A (reference 5).

In addition to the impact of the foregoing requirements on an airlines inspection/maintenance policy, the equipment manufacturers' designs presume an inspection/maintenance policy will be followed by the airlines using their equipment. For example, requirements for next generation fault tolerant electric engine controls

TABLE 4-1. EQUIPMENT REQUIRED FOR IFR DISPATCH
(FAR PARTS 25.1303 AND 121.303)

SYSTEM/SUBSYSTEM	NUMBER REQUIRED
AIR DATA	
STATIC PRESSURE SENSOR	2
FREE-AIR TEMPERATURE INDICATOR	1
AIRSPEED INDICATOR	1
ALTIMETER	2
VERTICAL SPEED INDICATOR	2
CLOCK WITH SWEEP SECOND HAND	1
ATTITUDE/HEADING REFERENCE	
MAGNETIC COMPASS	1
VERTICAL GYROSCOPE (ARTIFICIAL HORIZON)	2
VERTICAL GYROSCOPE - INDEPENDENT	1
SLIP-SKID (TURN-AND-BANK)	1
DIRECTIONAL GYROSCOPE	1
COMMUNICATION	
VHF VOICE (TWO-WAY)	2
VOR RECEIVER	2
DME RECEIVER	1
ILS RECEIVER	1
MARKER BEACON RECEIVER	1
ADF RECEIVER	1
AIRBORNE WEATHER RADAR*	1
AIRCRAFT LIGHTING	
INSTRUMENT LIGHTS	
POSITION LIGHTS	
ANTICOLLISION LIGHT	
LANDING LIGHTS	2
ELECTRICAL	
INDEPENDENT GENERATOR & DISTRIBUTION BATTERY	2
ENGINE INSTRUMENTS	
COCKPIT VOICE RECORDER	
LANDING GEAR AURAL WARNING	

*Must be operating satisfactorily if current weather reports indicate weather conditions that can be detected with airborne weather radar may reasonably be expected along route to be flown.

TABLE 4-2. AIRBORNE EQUIPMENT REQUIREMENTS
(CATEGORIES I AND II[†])

MINIMUM REQUIREMENTS	CAT I (TURBOJET ONLY)	CAT II (ALL AIRCRAFT)
SINGLE FLIGHT DIRECTOR* SINGLE AUTOMATIC APPROACH COUPLER**	REQUIRED	MINIMUM REQUIREMENT - TWO- ENGINE PROPELLER AIRCRAFT ONLY.
INSTRUMENT FAILURE WARNING SYSTEM	OPTIONAL***	REQUIRED PLUS FLIGHT CREW ASSIGNMENTS AND PROCEDURES SPECIFIED BELOW***
DUAL ILS AND GLIDE SLOPE RECEIVERS	NOT REQUIRED (N.R.)	REQUIRED
SINGLE FLIGHT DIRECTOR WITH DUAL DISPLAYS* AND SINGLE AUTOMATIC APPROACH COUPLER** OR TWO INDEPENDENT FLIGHT DIRECTOR SYSTEMS	N.R.	REQUIRED
EQUIPMENT FOR IDENTIFICATION OF DECISION HEIGHT	N.R.	REQUIRED. CAN BE: (1) RADAR ALTIMETER, OR (2) INNER MARKERS.
MISSED APPROACH ATTITUDE GUIDANCE	N.R.	REQUIRED. CAN BE: (1) ATTITUDE GYROS WITH CALIBRATED PITCH MARKINGS, OR (2) FLIGHT DIRECTOR PITCH (3) COMPUTED PITCH COMMAND.
AUTO THROTTLE SYSTEM	N.R.	REQUIRED ALL TURBOJETS IF OPER- ATIONS BASED ON DUAL FLIGHT DIRECTORS. ALSO REQUIRED ANY AIRCRAFT USING SPLIT AXIS COUPLERS IF APPLICANT CAN'T SHOW IT DOES NOT SIGNIFICANTLY REDUCE PILOT WORKLOAD.
RAIN REMOVAL EQUIPMENT	N.R.	REQUIRED

*SINGLE AXIS FLIGHT DIRECTORS IF BASIC GLIDE SLOPE INFORMATION
DISPLAYED ON SAME INSTRUMENT.

**SPLIT AXIS ACCEPTABLE.

***IF IMPROVED FAILURE WARNING SYSTEM NOT PROVIDED FOR CAT I OPERATIONS
APPLICANT MUST ESTABLISH FLIGHT CREW PROCEDURES AND DUTY ASSIGNMENTS
TO PROVIDE IMMEDIATE DETECTION OF ESSENTIAL INSTRUMENT AND EQUIPMENT
FAILURES. SUCH PROCEDURES AND ASSIGNMENTS ARE REQUIRED FOR CATEGORY
II OPERATIONS.

[†]Source: FAA ADVISORY CIRCULAR 120-29, SEPT. 25, 1970.

presume a class of faults for components that are inspected at 100-hour intervals, since the probability of creating a hazard to the safe operation of the aircraft is extremely low. These components might not be repaired for periods up to 100 flying hours after the fault occurs. This type of design concept does not necessarily presume the faults are detected immediately but presumes they are detected at the 100-flying hour inspection periods. Faults falling in this class of 100 hours inspection/detection are assumed to be repaired once they are detected.

The detection of faults through other than inspection at specific time periods is dictated by dispatch requirements and specific design practices. The different monitoring concepts for failures are addressed in section 5 of this handbook and in ARINC Report No. 415-2 (reference 6) for selected subsystems.

4.1.2 Airport Configuration.

Airport surface delays are due to time elapsed after departing the gate before reaching the takeoff runway, and after landing before shutting down the aircraft, once the gate is reached. In some analyses, the airport surface delays are ignored since loss of a critical function prior to actual takeoff should result in the aircraft returning to the gate for replacement/repair of the failed components, and loss of a critical function after touchdown and turnoff of the active runway would result in replacement/repair of the failed components, once the aircraft has been shutdown and before dispatch on the next flight leg. This may be influenced by the fault detection/annunciation capability of the digital flight control and avionics systems. If there is a high probability that failures of components required for critical functions are detected and the crew is alerted immediately upon detection, the assumption that airport surface delays can be ignored may be valid. If the system involved in the critical function has no self-test or other means of detecting that a fault has occurred, then the airport configuration impact on ground operating time should be taken into account.

4.1.3 Air Traffic Mix/Density at Specific Airports as a Function of Time-of-Day.

In addition to the consideration of the impact on system operating time of specific characteristics of an airport's configuration, the air traffic mix/density at the airport contributes to the terminal area delays due to airport capacity considerations, and air traffic control rules governing separation between different classes of aircraft. At peak traffic hours, the nominal operating time should have a factor added for airport delays due to the traffic.

4.1.4 Air Traffic Control/Navigation Aids.

The navigation aids available in the geographic area that the aircraft flies over between city pairs, and the number of aircraft simultaneously using the navigation aids, interact with the aircraft's avionics and influence the flight's nominal ground track, and hence the time required to fly between airports.

The primary radio navigation aids used are given in table 4-3. This data is taken from the Federal Radionavigation Plan (reference 7). The requirement for use of a VOR/DME in IFR operation, as well as the widespread coverage of this system, may undoubtedly result in increased use of area navigation. The time and fuel savings possible with direct routes, while significant from an operator's viewpoint, should not result in a change in the operating time used in the reliability and safety analyses performed during the validation of the digital system.

TABLE 4-3. PRIMARY RADIO NAVIGATION SYSTEM

<u>SYSTEM</u>	<u>COVERAGE</u>	<u>STATUS</u>
VOR	150 Stations in US	Operational Use Through 1995
VOR/DME VORTAC	770 Stations in US	Operational Use Through 1995 Operational Use Through 1995
TACAN	140 Ground Beacons	Phase Out from 1985-1995
ILS	581 Facilities - Glideslope + Localizer 54 Facilities - Localizer only	Operational
MLS	Gradual Replacement of ILS	Developmental
LORAN-C	Coastal Areas and 2/3 of Conterminous 48 States	Operational Use Through 2000
OMEGA	Worldwide	8 Stations Operational
NAVSTAR GPS	Worldwide When Operational in 1987.	Developmental

4.1.5 Operating Time Considerations in Validation.

The current trend toward stating required safety levels in terms of occurrences per hour of flight time does not provide sufficient guidance as to the specific time to use for the different phases of a flight, such as takeoff or landings, nor the total operating time for a single flight. While Advisory Circular 25.1309-1 recommends the use of fault trees in the analysis, it does not indicate that the result of the analysis is the probability that the top event exists at some time, i.e., time T. Those performing validation should use caution in reviewing analyses performed by others. In the construction of the fault tree, there is no explicit recognition of time. This can be overcome, at least in many cases, by time related definition of events. For example, a top event could be defined as loss of control during a specific period of time, such as final approach and landing. If there is more than one time period of interest, it may be necessary to construct a different fault tree for each time period, and for each top event in each time period (reference 8).

It is recommended that the analyst use a standard system operating profile for a single flight with the takeoff to landing time based on the maximum endurance of the aircraft. Table 4-4 presents an example of a standard system operating profile. During this time, the digital systems are always "ON" even though all functions may not be required in all flight phases. A failure can occur during this "not required" phase causing the system to be "NOT AVAILABLE" when the function is required.

TABLE 4-4. EXAMPLE OF DIGITAL FLIGHT CONTROL AND AVIONICS
SYSTEM OPERATING PROFILE

Estimated Time Duration	
Mission Phase	(Minutes)
Gate Departure to Takeoff	15
Takeoff	3
Climb-Out	8
Cruise	180
Holding	30
Let-Down	10
Final Approach/Landing	3
Taxi-to-Gate	5

4.2 ATMOSPHERIC ENVIRONMENT.

There are many factors in the atmospheric environment that influence the flight of an aircraft. All aspects of the weather including temperature, winds, clouds, fog, precipitation, and lightning influence the flight of an aircraft and the equipment it must carry if it is designed to operate in weather. Airline operations take place on a scheduled basis and airlines attempt, within the bounds of safety, to maintain the schedule in all weather conditions so long as the FAA and local airport authorities permit operation. To aircraft pilots, the acts of nature are something to avoid, if at all possible. While the pilot may avoid bad weather most of the time, the aircraft must be designed to fly safely through the weather if encounter is unavoidable.

Of the hazards noted, atmospheric electrical phenomena pose the most severe threat to aircraft which have digital avionics and flight control systems. There are two primary naturally occurring electrical phenomena that affect such systems; lightning and static electrification. In addition, the designer must be concerned with internally generated electromagnetic interference (EMI) interfering with the correct operation of the digital systems. Just as the pilot attempts to avoid weather, so should the designer attempt to avoid any interference phenomena that may impact the correct operation of the digital systems.

Interference control of sources internal and external to aircraft has become a field of its own. Historically, commercial airlines relied on military standards and other industry documents not necessarily directly related to airline applications. In 1975, the Radio Technical Commission for Aeronautics (RTCA) released DO-160, "Environmental Conditions and Test Procedures for Airborne Equipment" to set forth industry adopted standards and test procedures for the control of radio and electrical interference DO-160 was updated in January 1980 to DO-160A (reference 9).

DO-160A describes the amount of interference which a system must be able to tolerate and how much interference the system itself is allowed to generate. ARINC Report No. 413A (reference 10) describes, in addition to the basic aircraft electric power characteristics, detailed guidance setting forth the aircraft's needs for transient protection, interference control, and basic design considerations for both equipment and installations.

This section will deal with the related topics of how lightning and electromagnetic interference affect the operations of digital avionics.

4.2.1 Lightning.

Lightning and static electricity are naturally occurring phenomenon which pose a threat to present day aircraft. The two primary factors contributing to this concern are: (1) The increasing usage of microelectronics, both analog and digital, in all types of systems and subsystems; and (2) the reduced electromagnetic shielding offered by advanced structural materials used by modern day aircraft (reference 11). The former is discussed in detail below; the latter is not addressed.

Indirect effects present the most severe threat to microelectronic-based avionic equipment. Sources of indirect effects include: Electromagnetic fields caused by a direct lightning strike or nearby lightning and static electricity caused by precipitation such as ice or rain. These effects can result in voltage transients in internal aircraft wiring. Since microelectric circuits use much lower power levels than did earlier electronic components, they are more sensitive to transient overvoltages. The voltage levels of such transients may be sufficient to cause temporary or permanent damage to microelectronic equipment.

Designing adequate protection for microelectronics in aircraft does not follow any one method. There are numerous tradeoffs and options depending upon the criticality of the system being protected, aircraft performance, life-cycle considerations, and ease of implementation. For protection against direct effects, arc surge suppressors and shielded cables are available to limit the induced voltage. For indirect effects, use of isolation devices, shielding, suppression, and terminal protection devices is recommended. Another philosophy in the handling of indirect effects include redundancy, and coding and signal processing techniques for error detection and/or correction (reference 10). Table 4-5 summarizes indirect effects protection from reference 11. Additional information may be obtained from references 12-18 and from the extensive research in lightning performed for the Space Shuttle program.

TABLE 4-5. PROTECTION FROM INDIRECT ELECTROMAGNETIC EFFECTS

Locate Equipment Away From Apertures

Locate Wiring

- o Away from apertures
- o Away from regions where the radius of curvature of structural members or outer skin is smallest (see figures 4-1 and 4-2)

Shielding

- o Ground at both ends
- o Multiple grounds will limit effect to that length between grounds

Use Narrow Bandwidth Components

- o Most of the energy associated with lightning is below 10-20 KHz
- o Lightning energy excites oscillatory frequencies on wiring in the range of several hundred kilohertz to a few megahertz.
- o Use of fiber optics, which uses bandwidth in the infrared region, avoids the area of lightning generated interference almost completely (see figure 4-3).

Use Resistor or Transformer Coupling to Semiconductors

Use Suppressors

- o Virtually eliminates physical damage but does not eliminate interference.
- o Types
 - Spark gaps
 - Zener Diodes
 - Varistors
 - Dielectrics

4.3 ELECTROMAGNETIC INTERFERENCE.

Electromagnetic interference (EMI) can be defined as any unplanned electrical signal that causes degradation or malfunction of system performance. The reduction and control of EMI so that a system can operate in a given EMI environment and not produce excessive EMI levels itself, is referred to as electromagnetic compatibility (EMC).

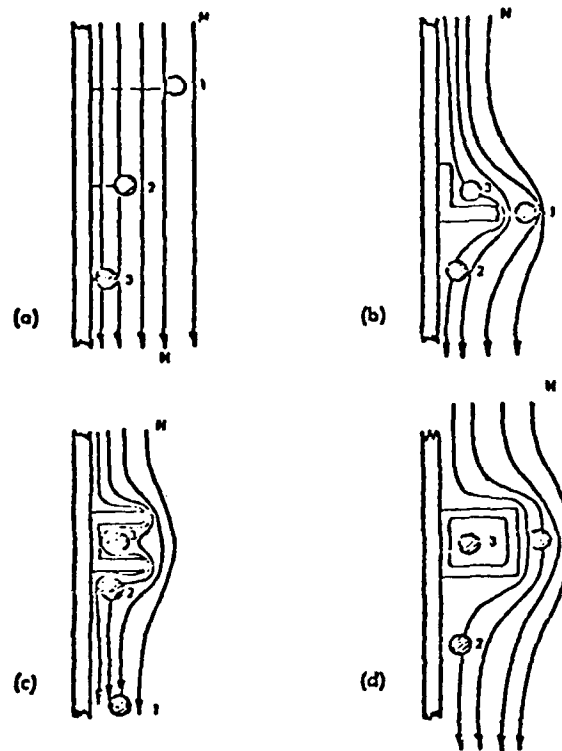
Table 4-6 lists sources of externally and internally generated EMI for a hypothetical digital avionic subsystem and components inside the subsystem susceptible to EMI (references 17, 19, 20).

EMI is characterized by extremely wide bandwidth. New microelectronics require more bandwidth than past electronics, therefore making microelectronics more susceptible to EMI.

Table 4-7 lists ways of improving the EMC of a given system (references 9, 10, 20-22).

4.4 SAFETY REQUIREMENTS.

The mission safety requirements are specified by the FAA and the British Civil Aviation Authority in terms of ranges of probability for which it must be shown that a given event will not occur. The consequences of the occurrence of the event and the contributors to its occurrence including failures due to lack of reliability or accuracy must take into account in the analysis. Furthermore, the analysis must take into account the environmental conditions, the functions required for a

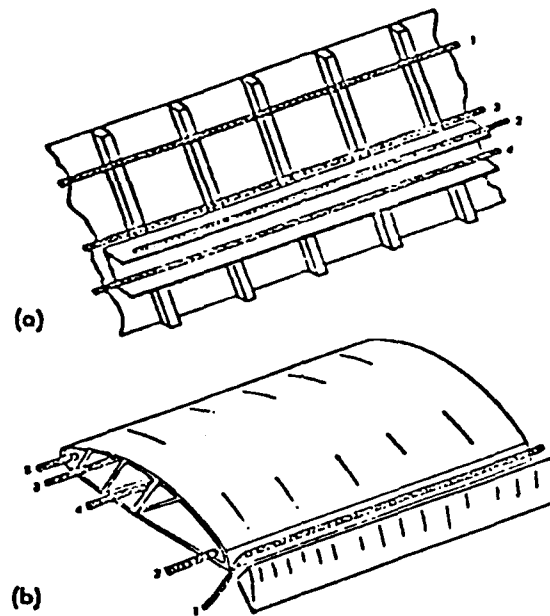


- (a) Conductors over a plane
- (b) Conductors near an angle
- (c) Conductors near a channel
- (d) Conductors near a box.

In each case pictured

Conductor 1 - highest flux linkages: worst
 Conductor 2 - intermediate linkages: better
 Conductor 3 - lowest linkages: best

FIGURE 4-1. FLUX LINKAGES VERSUS CONDUCTOR POSITION (Reference 12)



- (a) A fuselage structure
 (b) A wing structure

In each case pictured

Conductor 1 - highest flux linkages: worst
 Conductor 4 - lowest flux linkages: best

FIGURE 4-2. CONDUCTOR ROUTING (Reference 12)

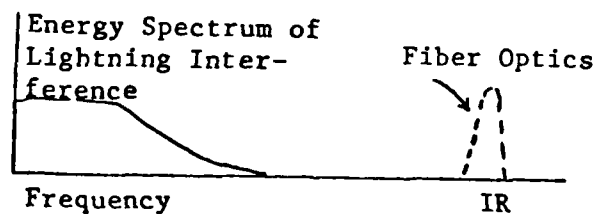


FIGURE 4-3. FREQUENCY CONSIDERATIONS

specific flight phase, and the exposure time. The FAA, in Advisory Circular 25.1309-1 (reference 23) and the British Civil Aviation Authority in references 24 and 25, basically require that events fall into the same range of probabilities, with the CAA subdividing the probable and improbable events as shown in table 4-8. The CAA further classifies the effect of the occurrence as minor, major, hazardous, or catastrophic as shown in figure 4-4, reproduced from reference 24.

TABLE 4-6. EMI SOURCES AND SUSCEPTIBILITY OF A
HYPOTHETICAL DIGITAL AVIONICS SUBSYSTEM

EMI SOURCES

External to the Subsystem

Atmospheric

- Lightning
- Solar Noise
- Cosmic Radiation
- Precipitation Static

Man-Made

- Switching Power Supplies
- Relays
- Motors
- Solenoids
- Transformers
- Cross-talk in Wiring
- Radar
- Communication Systems such as (2-30 MHz)
 - Radio Transmitters (200-450 MHz and 1 to 40 GHz)
- Altimeters
- Transponders
- Navigation Aids (300-500 KHz)
- Vibration

Internal to the Subsystem

- Oscillators such as Clock Generators
- Digital Logic--(although low relative to other sources)
- Signal Reflections
- Cross-talk

Susceptible Components of the Subsystem

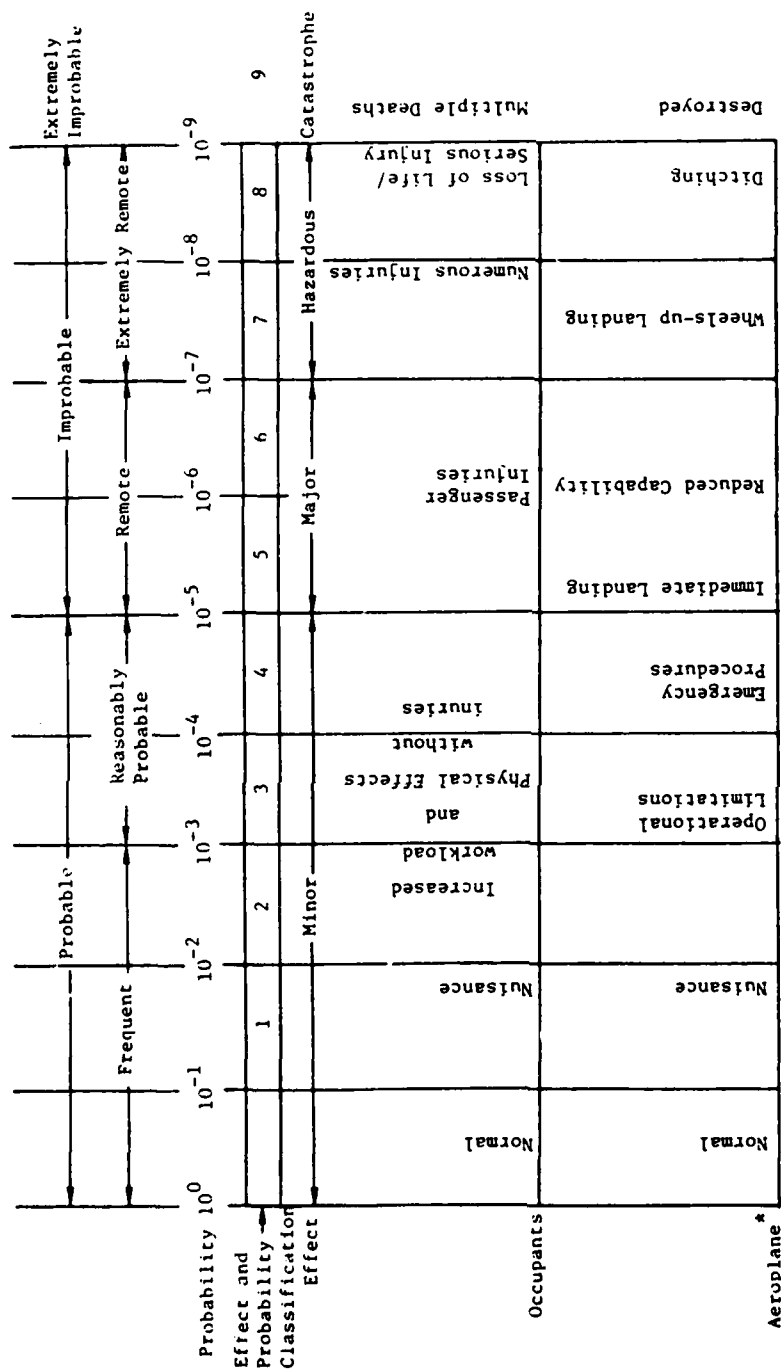
- A/D Input Circuits
- Signal Conditioning Circuits
- Amplifiers
- Cabling
- Digital Logic
- Openings in Box or Apertures
- Secondary Coupling through Box to Circuitry
- Cathode Ray Tubes (CRT)
- Radar Screen

TABLE 4-7. HOW TO IMPROVE EMC OF THE SUBSYSTEM

- o Use EMI filters on a.c. powerlines
- o Use cable shielding grounded at both ends
- o Identify a preferred structural path as a continuous metallic path from the preferred entry to the preferred exit point for lightning
- o Use of metal skinned a.c. good shield for EMI compared (d.c. to 1 GHz) to nonmetal skinned a.c. (see discussion on lightning)
- o Component placement away from high exposure areas
- o Do not use airframe as a signal return for electrical wiring
- o Use conductive epoxies, greases, paints
- o Use EMI gaskets between joints (conductive elastomers)
- o Plan wire routing control
 - Physical separation into bundles of similar characteristics is good
 - Power wiring
 - Secondary power
 - Control wiring
 - Sensitive wiring
 - Susceptible wiring
 - System wiring
- o Shielding
 - Special shielding materials must still maintain adequate thermal transfer
 - Air filters, screen or honeycomb -- opaque to EMI
 - Absorbs and reflects energy
 - Ability to conduct electric and magnetic fields

TABLE 4-8. QUANTITATIVE SAFETY REQUIREMENTS

Frequency of Occurrence	Probability Range		
	(Events per hour)		(Events per Flight)
	FAA (Ref. 23)	CAA (Ref. 24,25)	USAF (Ref. 28)
Probable	$\leq 10^{-5}$	10^0 to 10^{-5}	
Frequent		10^0 to 10^{-3}	
Reasonably Probable		10^{-3} to 10^{-5}	
Improbable	10^{-5} to 10^{-9}	10^{-5} to 10^{-9}	
Remote		10^{-5} to 10^{-7}	
Extremely Remote		10^{-7} to 10^{-9}	$\leq 5 \times 10^{-7}$
Extremely Improbable	less than 10^{-9}	less than 10^{-9}	



* The positions of some of the items on this scale will vary for different aeroplane designs.

FIGURE 4-4. RELATIONSHIP BETWEEN PROBABILITY AND SEVERITY OF EFFECTS (Reference 29)

Both the FAA and the CAA further require specific performance accuracy as well as safety for functions such as autoland (references 4, 5, 26 and 27). The performance requirements are specified in terms of glide slope and localizer deviations during Category I, II, and III approaches and for along-track and cross-track dispersions at touchdown in autoland.

The CAA describes safety in terms of average risk and specific risk in autoland. Average risk is expressed in terms of the total fatal landing accident rate. In particular, the autoland system must have reliability and performance such that the probability of a fatal landing accident due to the autoland function is less than or equal to 1×10^{-7} . Specific risk refers to the risk associated with using the autoland system on a specific flight. The maximum risk of using the system should be less than the total average risk for a complete flight. This should be measured at the last point in the flight from which a safer diversion can be made.

For a given incident, risk is defined as a product of the probability the incident occurs and the probability of the fatal accident given the incident occurs. Incidents can be related to failures (e.g., computer malfunction) or poor performance (e.g., large cross-track deviation at touchdown).

Risk assessment is required to include all incidents involving the autoland system which could result in a fatal accident. Factors to be considered include the following:

- (1) Crew performance.
- (2) Component and system failures.
- (3) Statistical variation in component and system performance.
- (4) Weather.

For certain incidents, estimates of fatal accident probabilities are provided based on past experience and assuming a defined airport configuration. The definition of average risk implicitly requires consideration of two factors:

- (1) State of the autoland system at flight initiation.
- (2) Duration of flight prior to use of the autoland system which corresponds to the exposure time defined in Advisory Circular 25.1309-1 (reference 23).

The initial state of the system is a function of the frequency and accuracy (i.e., coverage) of autoland inspection tests. In order to analyze an actual autoland system, it may be necessary to develop simplifying assumptions to model the tests in a manner in which they can be supported from available data and engineering estimates. An upper bound for flight duration can be used to develop a "worst case" estimate of undetected autoland system failure prior to initiation of automatic landing.

It should be noted that the CAA states "the probability should be established as a risk per hour in a flight where the duration is equal to the expected mean flight time and for the airplane. For example, in systems where the hazard results from multiple failures in the same flight, the numerical assessment should take account of the likelihood that this will occur in a flight of expected average duration. Similarly, in those cases where failures are only critical for a particular period of flight, the hazard may be averaged over the whole of the expected mean flight time" (reference 24).

The FAA states in reference 23 "reliability study determines the probability of the single faults used as bottom level events of the fault tree from component failure rate data and exposure times to both active and latent failures. The probability of all event conditions in the fault tree will then be calculated from these data. The fact that maintenance or flight crew checks will be performed throughout the life of the system is relevant to quantitative analysis. When exposure times, relevant to failure probability calculations, are affected by flight crew checks or inspection intervals, these time intervals shall be clearly specified in the appropriate documents."

4.5 CONSTRAINTS.

A number of things involved in operation of an airline act as constraining factors. The constraining parameters include those related to the aircraft, the airport, approach/landing performance limits, weather, and airline imposed constraints. These are discussed in the following paragraphs.

4.5.1 Aircraft.

The physical aircraft constraints include: (1) Operating empty weight; (2) maximum fuel capacity; (3) maximum gross takeoff weight; (4) maximum gross landing weight; (5) maximum takeoff thrust; (6) physical dimensions including wingspan, length, height, and gear footprint; (7) aircraft operating ceiling; (8) aircraft performance including rate of climb, rate of sink, braking, etc.; and (9) real-world realizable MTBF and maintainability parameters for onboard systems including avionics.

4.5.2 Airport.

The physical configuration of the airport is a significant constraining factor in that the configuration has a significant influence on the number of operations that may be conducted per hour at that airport under various weather conditions for the air traffic mix and density utilizing that airport. In addition, the location of the airport in itself effectively creates a constraint due to weather conditions in that locality. The number of scheduled and unscheduled operations performed at the airport as a function of time-of-day also act as a constraint. The combination of foregoing factors ultimately determines the delays to be expected by the aircraft because of the foregoing airport related parameters.

4.5.3 Approach/Landing Performance Limits.

Realistic performance limits or constraints during the approach and landing phase are described below. If the aircraft path and time errors exceed those limits, a missed approach should be executed. There are several reasons for choosing, or being directed, to execute a missed approach:

(1) If the terminal airspace is tightly controlled and the aircraft is expected to fly a specific path, then large deviations could create a collision hazard with other aircraft.

(2) If the traffic control has sophisticated sequencing and metering capabilities, then the aircraft is expected to maintain precise time control. Excessive deviation is cause for directing a missed approach.

AD-A133 222

VALIDATION OF DIGITAL SYSTEMS IN AVIONICS AND FLIGHT
CONTROL APPLICATIONS. (U) BATTELLE COLUMBUS LABS OH
E F HITT ET AL. JUL 83 DOT/FAA/CT-82/115

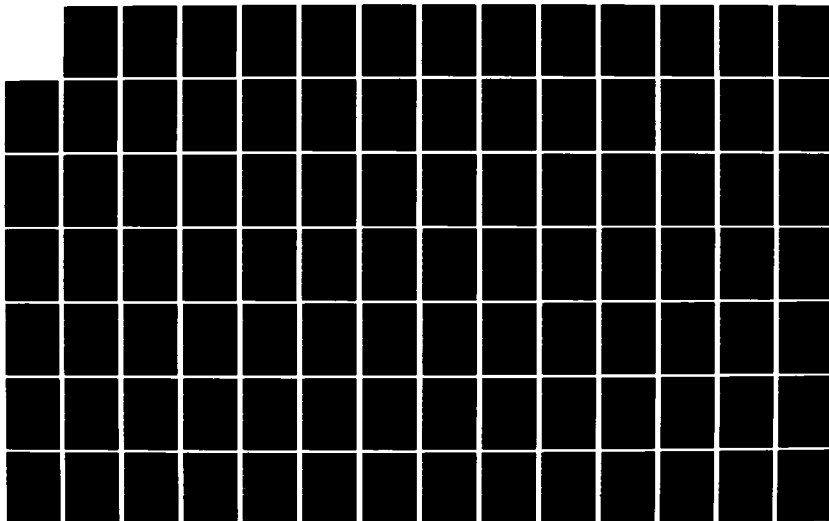
2/9

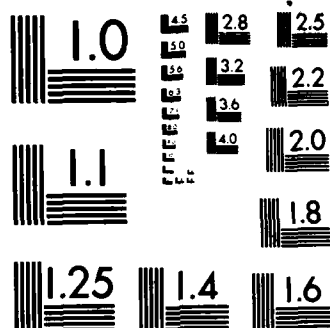
UNCLASSIFIED

DTFA03-81-C-00059

F/G 1/3

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(3) The most obvious reason for executing a missed approach is if the lateral or vertical performance in the last few hundred feet of altitude is such that an unsafe landing could result.

4.5.3.1 Causes of Excessive Deviations. Excessive deviations result from two sources: System failures and "poor" fault-free performance. The pilot can fit into either category with blunders appearing like failures and sloppy tracking looking like poor performance. Failures in critical system elements that are detected will cause a missed approach at the time of detection. Undetected failures will cause go-around when the adverse performance becomes obvious. It should be noted that before an aircraft-control system can be certified it must be shown that detected and undetected failures yield only a very remote probability of hazard (on the order of 10^{-7} to 10^{-9}). For Category I and II type landing systems, this remote probability is provided by insisting on pilot visibility for the final landing stage. For Category IIIA, it is through multiple redundant systems. By the same token, it must be assumed that the same assurance of remote probability of hazard due to failures will be provided for all operations in the terminal area. (It should be noted that some terminal area avionics may fail prior to reaching the terminal area but go undetected until used. To account for these cases, failure exposure time should be total flight time, but a portion of the terminal avionics equipment failures reflecting undetected failures should be treated as if they occur in the terminal area even though they occurred earlier.)

"Poor" fault-free performance (poor is in quotes because it does not necessarily reflect poor design or pilot performance) results, generally, from noisy state estimation, large external disturbances, or excess maneuvering requirements. Noisy state estimation is a matter-of-fact with present ILS and terminal VOR DME's. The new microwave landing system promises to relieve most of these problems. External disturbances include steady wind, turbulence, and wind shear. Large deviations can also occur if the turn onto the final course is close enough to touchdown that localizer capture overshoots have insufficient time to settle.

It is this "poor" performance which must be monitored in some manner to determine whether a missed approach is warranted.

4.5.3.2 Present Performance Monitoring Methods. Performance in the approach/landing phase is presently monitored as follows. For a Category I or II landing, the monitoring is with surveillance radar in the horizontal plane, with localizer deviation aboard the aircraft and with pilot visibility at the decision height. The surveillance radar is too crude to provide much more than gross monitoring. The pilot is controlling with the localizer deviation and he monitors that deviation to avoid large errors. (It should be noted that he smooths what he observes when the signal is noisy.) For example, when another aircraft overflies the localizer, it often causes large oscillatory deviations and in some cases radio flag warnings. However, the pilot recognizes these symptoms for what they are and smooths them unless he is close to the decision height. Precision monitoring is accomplished at the decision height when the pilot must visually verify that he can safely land or execute a missed approach.

For a Category IIIA landing, reliance is placed on the integrity of the control system and the ground navigation aids. If the pilot notes that the control signals are sufficiently nulled and there are no failure indications in either the aircraft or ground equipment, the landing proceeds. There are variations in how the integrity monitoring is performed. For example, the DC-10 used an independent

monitor which models the aircraft and control system and projects the landing point. The L-1011 uses dual/dual redundant systems and verifies that all systems agree. In addition, excess ILS deviation monitors are activated as the aircraft approaches touchdown. The purpose of both techniques is to detect failures and look for gross control errors.

Prior to reaching the final landing path, monitoring is provided with the surveillance radar and whatever navigation equipment is being used onboard the aircraft.

4.5.4 Airline Imposed Constraints.

While the allowable time after scheduled gate departure before delay is a variable, the airframe manufacturer might view that time as a constraint. This time is normally determined by the airline operator. In addition, the elapsed time past scheduled gate departure, at which time a decision to cancel a flight is reached, may be considered a constraint.

4.6 ADVANCED DIGITAL INTEGRATED FLIGHT CONTROL AND AVIONICS SYSTEM FUNCTIONS.

An advanced digital integrated flight control and avionics system makes use of digital computer technology, digital data communications, and software engineering to integrate navigation, guidance, communications, and energy management functions with the aircraft control functions (outer and inner loops). While these same functions have been performed since the beginning of aviation, it is only recently with the progression from no computers, to analog, and now to digital computers that it has become possible to integrate the functions efficiently and reduce the duplication of sensors and processing.

4.6.1 Flight Control Background.

Closed loop flight control has been implemented since the beginning of aviation through pilot inputs to mechanical devices (stick, rudder pedals, flaps controller, and throttle) which were directly linked to the appropriate control surfaces or actuators (elevator, ailerons, rudders, flaps, and engine) and controlled the position of these devices. As the control forces required exceeded "comfortable" force levels, and the aircraft's stability and handling qualities became less than acceptable, sensors were added and the pilot's inputs were augmented with electro-mechanical or hydraulic amplification to control the stability and flight trajectory of the aircraft. The progression of technology now permits removal of the direct mechanical linkages with the manual (pilot) or automatic control inputs to the actuators transmitted by electrical signals (analog or digital), which is referred to as fly-by-wire (FBW) (references 29-31).

4.6.2 Command Generation (Guidance) Background.

The capability to generate commands that permit control of the aircraft's motion along a desired flight trajectory has also progressed with computer technology. Early aircraft instruments sensed and displayed present magnetic heading, altitude, and airspeed. The determination of position, as well as heading and altitude changes, and the time to initiate the change needed to control the aircraft's flight along a desired trajectory was the responsibility of the crew. The advent of radio navigation aids assisted crewmembers in not only determining present position but also the angular (cross-track) deviation (displayed on the course deviation indicator (CDI) and horizontal situation indicator (HSI) and hence the

commands to correct the ground track to the desired radial. These measurements were processed in the flight director computer and a flight director symbol eventually was added to the attitude indicator which became an attitude director indicator (ADI). The displayed deviations and commands permitted the crew to manually control the present position vector (latitude, longitude, and altitude) and hence null the deviations from the nominal flight trajectory. The computed deviations could also provide needed inputs to the autopilot for automatic control of the outer loop. Further advances in computer technology have provided the memory and throughput needed to implement algorithms that command control of time of arrival as well as position vector (referred to by many as four dimensional (4D) guidance) (references 32-35). It is possible to implement algorithms which control the position and velocity vectors as well as time. This concept has significant implications for either manual or automatic control of flight. While position changes have previously been effected through attitude and thrust, recent advances in flight control such as direct lift control (DLC) (reference 36) permit wings-level lateral or vertical position translation. The implementation of the more complex algorithms for command generation with the multiple methods becoming available for closure of the outer control loop necessitates research on alternative integration concepts including man-machine task allocation.

Many studies (references 32, 33, 37, and 38) directly related to the development, analysis, and simulation of specific guidance (steering) algorithms have been performed. These studies include:

- (1) Automatic multisensor navigation employing a Kalman filter.
- (2) Data link input of commanded heading, altitude, and Mach; time-to-go; range, bearing, and altitude, etc.
- (3) Generation of guidance (steering) commands for positioning of command display symbology or input to automatic control (autopilot) system to direct flight along a nominal reference trajectory that may be described as:
 - (a) Two dimensional (2D) — latitude, longitude.
 - (b) Three dimension (3D) — latitude, longitude, altitude.
 - (c) Four dimensional (4D) — 3D and specific time.
 - (d) Four state — 3D and heading.
 - (e) Five state — 3D, heading and specific time.
 - (f) Six state — 3D, groundspeed, heading, and specific time.
 - (g) Combinations of the above.
- (4) Throttle/energy management.

Each of the foregoing has included various levels of integration, but most guidance (steering) commands are designed to either: (1) Control to or track the nominal flight trajectory (which may be modified by the crew through manual input or as a result of data link input(s); or (2) control to the next waypoint in the sequence.

4.6.3 State Estimation (Navigation) Background.

Navigation has progressively achieved higher levels of integration with the move from the crew observing known landmarks, to automatic multisensor navigation. This latter concept uses a Kalman filter, implemented in the digital computer, to weigh the measurements provided by the various sensors and arrive at a best

estimate of the vehicle state (e.g., latitude, longitude, altitude, velocity, acceleration, attitude, and angular rates) at the current time with respect to the airmass and ground referenced coordinate frame.

4.6.4 Aircraft System Functions.

As the foregoing background indicates, integrated control has been under development for many years. Integrated control is essentially:

- (1) Processing the measurements from multiple sensors to derive a best estimate of the current (present time) vehicle state.
- (2) Computing the difference between the nominal (desired) trajectory state and estimate of current vehicle state.
- (3) Computing, using the computed difference:
 - (a) The appropriate steering command for positioning of command symbology displayed to the crew.
 - (b) Inputs to the automatic flight control system for inner and outer loop control modes selected by the crew.

These functions are the basic aircraft system functions which comprise integrated control.

An aircraft system function is defined as an operation or action required to conduct the mission. Integrated control system functions include systems management, state estimation (navigation), command generation (guidance), command execution (control), communications, energy management, and fuel management. A particular function, such as the control function, may be done automatically, manually, or by a combination of automatic and manual processes. The combinations of processes for a particular function shall be defined as specific subfunctions. Each subfunction may be performed in a variety of modes.

A mode is defined as the specific set of measurements (inputs) and appropriate algorithms which are processed to provide desired outputs. These measurements may be provided by specific combinations of sensor hardware and software, or may make use of optimal estimation theory (such as the Kalman filter) to compute or estimate the measurements based on inputs from sensors which may provide redundant measurements or information. This latter technique has been referred to as analytic redundancy (references 39-44) and provides a continuous estimate of the measurements in the presence of permanent or partial failures of some sensors. The specific mode that is utilized in performing a function is dependent upon the hierarchy of modes established by the system designers for automatic subfunctions and the crew selection of modes using manual subfunctions as well as the status of the system where status is defined as the hardware/software vector of information regarding all hardware and software conditions at the present time.

4.6.4.1 Navigation, Guidance, and Control Functions. If the state estimation (navigation) function is done automatically, defined as the subfunction of automatic navigation, the onboard processors (computers) would sample a variety of state measurements, as shown in figure 4-5, provided either sequentially, if a serial digital interface is used, or simultaneously if a parallel interface is

used. If a particular state measurement is unreliable due to measurement uncertainties or hardware subsystem failures, the software logic should cause reversion to a backup mode (often of lower performance) and process only those measurements available in automatic navigation. Crew selection of state estimation modes requires various manual navigation subfunctions to be defined. Whatever the function, subfunctions permit automatic or manual modes.

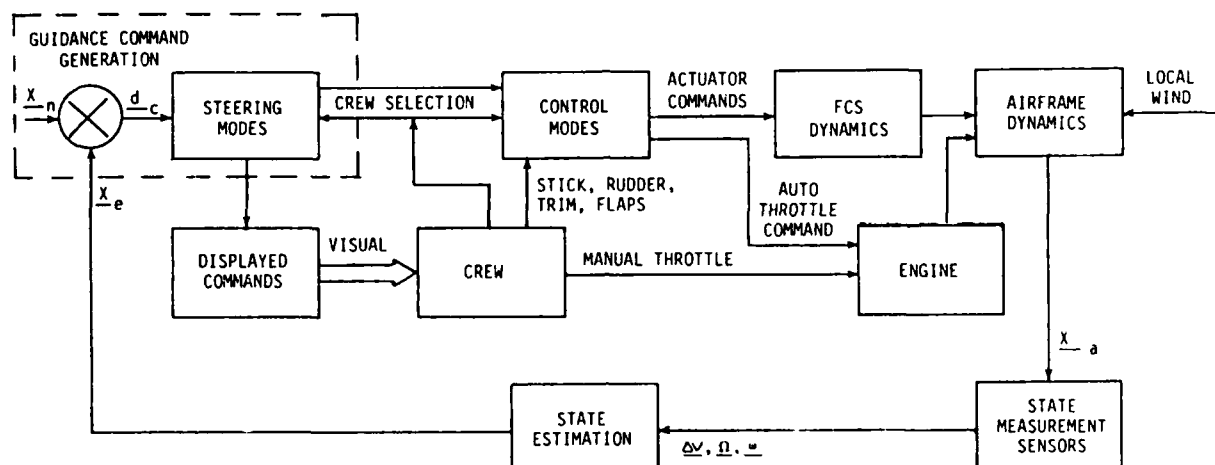


FIGURE 4-5. BLOCK DIAGRAM OF NAVIGATION, GUIDANCE, AND CONTROL OUTER LOOP

The basic selection of control modes (inner and outer loop) and steering modes is done by the crew. The system designer must develop the logic which prohibits engaging incompatible inner and outer loop modes. The crew may elect to manually control all outer loop axes based upon displayed commands while the inner loop is automatically controlled. The crew may also elect to control only certain outer loop axes while the others are automatically controlled. For example, the crew may select automatic control of altitude and manual control heading.

As can be seen, the guidance function computes a deviation vector (\underline{d}_c) which is primary input to the steering and control algorithms. The \underline{d}_c vector may be computed onboard or may be derived from tracking radars operated by air traffic control. Externally computed \underline{d}_c may be data linked to the aircraft or communicated by controllers over voice frequencies (channels). In integrated control, all components of the \underline{d}_c vector are continuously computed. If the crew selects a manual subfunction for guidance or control, this also requires selection of the specific manual mode desired. It should be recognized that the selection of the automatic or manual subfunction is normally a function of flight phase (i.e., takeoff, cruise, descent, and landing).

4.6.4.2 Systems Management Function. Digital systems such as the integrated control system utilize a digital communications architecture made up of computers, digital data transfer channels, and the associated sensors and controllers required to implement the control process. The primary elements which must be managed are the computers and the digital data transfer channels. These may be considered to make up a network of computers interconnected via the digital data transfer channels. The management of this network as well as the other integrated control functions comprises the primary system management function. The system management function may be implemented with both hardware and software. The software must implement the system management operating system (executive software for control of communication protocol which includes software implemented error detection and recovery) and interface with the local executive in each processor. The performance of the system management function may be considered analogous to a large extent to that of digital communication systems including such parameters as system and channel capacity, message errors, etc. The system management function's performance is critical to the successful implementation of integrated control.

Systems architectures such as the triple redundant mechanizations, with self-test and in-line monitoring, are representative of the concepts for implementing a fault tolerant system management function in conjunction with the flight control function. The use of majority voting exemplifies present concepts for avoidance of hardware and software faults.

4.6.4.3 Communications Function. The communications function includes four subfunctions which involve either transmitting, receiving, or both, of voice and digital data. These communications subfunctions do not include state measurement sensors which receive, or transmit and receive, data used in computing the current vehicle state (such as from VOR or DME). The communication subfunctions utilize transceivers that operate in bands such as:

- (1) VHF.
- (2) UHF.
- (3) HF.

The communications subfunctions include:

- (1) UHF.
- (2) VHF/FM.
- (3) Transponder.
- (4) Data Link.

4.6.4.4 Energy Management

The energy management function consists of four subfunctions (reference 37):

- (1) Fuel/time tradeoff.
- (2) Energy management steering.
- (3) Autothrottle.
- (4) In-flight calibration.

Measurements required are given in reference 37.

4.6.4.5 Time Dependency of Aircraft System Functions. Primary functions required in all flight phases, which mainly involve change of subfunction from automatic to manual and vice versa as a function of flight phase are:

- (1) State Estimation (navigation).
- (2) Command Generation (guidance/steering).
- (3) Command Execution (control).

The digital systems management function must also operate continuously during all flight phases. The communications function is utilized at specific times, determined by the need to communicate, during all flight phases. The energy management function may be used in all flight phases.

4.7 REFERENCES.

1. Federal Aviation Regulations, Volume III, Part 25, Department of Transportation, Federal Aviation Administration, December 1969.
2. Federal Aviation Regulations, Volume VII, Part 121, Department of Transportation, Federal Aviation Administration, June 1970.
3. En Route Air Traffic Control, Handbook 7110.9A, Department of Transportation, Federal Aviation Administration, Air Traffic Service, January 1, 1970.
4. Criteria for Approval of Category III, Landing Weather Minima, AC 120-28C, Department of Transportation, Federal Aviation Administration, May 13, 1982.
5. Automatic Landing Systems (ALS), AC 20-57A, Department of Transportation, Federal Aviation Administration, January 12, 1971.
6. Operational and Technical Guidelines on Failure Warning and Functional Test, ARINC Report No. 415-2, ATA Report No. 112-2, Aeronautical Radio, Incorporated, January 10, 1968.
7. Federal Radionavigation Plan, Department of Defense and Department of Transportation, July 1980.
8. Hitt, E. F., Bridgman, M. S., and Robinson, A. C., Comparative Analysis of Techniques for Evaluating the Effectiveness of Aircraft Computing Systems, NASA CR 159358, Battelle-Columbus Laboratories, April 1981.
9. Environmental Conditions and Test Procedures for Airborne Equipment, Document No. RTCA/DO-160A, Radio Technical Commission for Aeronautics, January 1980.
10. Guidance for Aircraft Electrical Power Utilization and Transient Protection, ARINC Report 413A, Aeronautical Radio, Incorporated, December 30, 1976.
11. DuBro, Gary A., Atmospheric Electricity Interactions with Aircraft: An Overview, AGARD Lecture Series No. 110, Atmospheric Electricity-Aircraft Interaction, pp. 1-1 through 1-11 May 1980.
12. Plumer, J. Anderson, Protection of Aircraft Avionics from Lightning Indirect Effects, *ibid*, pp. 4-1 through 4-27, May 1980.

13. Fisher, Franklin A. and Plumer, J. Anderson, Lightning Protection of Aircraft, NASA Reference Publication 1008, National Aeronautics and Space Administration, 1977.
14. Plumer, J. A., Fisher, F. A., and Walko, L. C., Lightning Effect on the NASA F-8 Digital Fly-by-Wire Airplane, NASA Contractor Report CR-2524, General Electric, March 1975.
15. Test Waveforms and Techniques for Assessing the Effects of Lightning-Induced Transients, SAE AE4L Committee Report: AE4L-82-2, December 15, 1981.
16. LeVine, D. M., Lightning Electric Field Measurement which Correlate with Strikes to the NASA F-106B Aircraft, NASA Technical Memorandum TM-82142, May 1981.
17. Reliability Advancement for Electronic Engine Controllers, Technical Report AFWAL-TR-80-2063, Hamilton Standard Division of United Technologies Corporation, May 1981, Pages 185-200.
18. Bonding, Electrical, and Lightning Protection for Aerospace Systems, Military Specification MIL-B-5087B, February 6, 1968.
19. Deavenport, Joe E., EMI Susceptibility Testing of Computer Systems, Computer Design, March 1980.
20. Howell, Dave, EMI Can Be Licked, Electronic Products, March 1978.
21. Matisoff, Bernard S., Good Shielding Techniques Control EMI and RFI, EDN, February 18, 1981.
22. King, G. J., Achieving Electromagnetic Compatibility by Control of the Wiring Installation, presented to Ninth Tri-Service Conference on Electromagnetic Compatibility, Douglas Paper No. 1661, October 1963.
23. Airplane System Design Analysis, Advisory Circular 25.1309-1, Department of Transportation, Federal Aviation Administration, Federal Register, Vol. 46, Issue 214, Pages 54958, September 7, 1982.
24. The Safety Assessment of Systems, Subsection D1-General and Definitions British Civil Airworthiness Requirements, Civil Aviation Authority, December 16, 1981.
25. The Certification of Safety Critical Digital Systems, CAA Airworthiness Information Leaflet, Civil Aviation Authority, April 25, 1978.
26. Airworthiness Requirements for Automatic Landing Including Automatic Landing in Restricted Visibility Down to Category III, British Civil Airworthiness Requirements Paper No. 367, Issue 3, Section D Aeroplanes, Air Registration Board, Civil Aviation Authority, June 1970.
27. Criteria for Approving Category I and Category II Landing Minima for FAR 121 Operators, Advisory Circular 120-29, Department of Transportation, Federal Aviation Administration, September 25, 1979.

28. Flight Control Systems-Design, Installation and Test of Piloted Aircraft, General Specification for, MIL-F-9490D (USAF), June 6, 1965.
29. Hooker, David S., et al., Survivable Flight Control System Interim Report No. 1, Studies, Analyses, and Approach, AFFDL-TR-71-20, McDonnell Aircraft Company, May 1971.
30. Kisslinger, Robert L., and Lorenzetti, Major Robert C., The Fly-by-Wire Systems Approach to Aircraft Flying Qualities, MCAIR 72-007, McDonnell Aircraft Company, 1972.
31. Seacord, C. L., and Vaughn, D. K., Preliminary System Design Study for a Digital Fly-by-Wire Flight Control System for an F-8C Aircraft, NASA CR-2609, Honeywell, Incorporated, January 1976.
32. Bird, Michael, Feasibility Study for Integrated Flight Trajectory Control (Airlift), AFFDL-TR-77-120, Lear Siegler, Incorporated, Instrument Division, January 1977.
33. Comegys, Gregory L., Feasibility Study for Integrated Flight Trajectory Control, AFFDL-TR-XX-SSS, Lear Siegler, Incorporated, Instrument Division, June 1979.
34. Erzberger, H., and Pecsvaradi, T., 4-D Guidance System Design with Application to STOL Air Traffic Control, presented at the 1972 Joint Automatic Control Conference, Paper 14-1, Stanford University, Stanford, CA., August 1972.
35. Foudriat, Edwin C., Aircraft 4-D Constant Velocity Control System, Journal of Aircraft, Vol. 11, No. 6, Pages 326-333, June 1974.
36. Wendl, M. J., Additional Degrees of Freedom, AGARD Lecture Series No. 89, TASK-Oriented Flight Control Systems, June 1977.
37. Marino, A. S., McAdam, W. E., and Chojnacki, R. C., Definition and Analysis of Flight Command Functions and Computations, AFFDL-TR-68-145, Hughes Aircraft Company, November 1968.
38. Stephan, P. W., and Chandler, W. J., Simulation Evaluation of Flight Command Functions, AFFDL-TR-70-4, Hughes Aircraft Company, March 1970.
39. Poyneer, Robert D., and Cunningham, Thas B., Fault Tolerant Digital Flight Control Using Analytic Redundancy, NARCON '77 Record, Pages 111-120.
40. Shapiro, E. Y., Software Techniques for Sensor Redundancy Management of Flight Control Systems, Journal of Aircraft, Vol. 14, No. 7, Pages 632-638, July 1977.
41. Cunningham, T. B., et al, Fault Tolerant Digital Flight Control with Analytical Redundancy, AFFDL-TR-77-25, Honeywell, Incorporated, May 1977.
42. Clark, R. N., Fosth, D. C., and Walton, V. M., Detecting Instrument Malfunctions in Control Systems, IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-11, No. 4, Pages 465-473, July 1975.

43. Clark, R. N., Instrument Fault Detection, IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-14, No. 3, Pages 456-465, May 1978.

44. Clark, R. N., A Simplified Instrument Failure Detection Scheme, IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-14, No. 4, Pages 558-563, July 1978.

SECTION FIVE
SYSTEM ARCHITECTURES

TABLE OF CONTENTS

	Page
SECTION 5	5-1
5. SYSTEM ARCHITECTURES	5-1
5.1 Fault Tolerant Digital Integrated Flight Control and Avionics	5-1
5.1.1 Redundancy Techniques	5-4
5.1.2 Motivation for Fault Tolerant Integrated Control	5-5
5.2 Advanced Integrated Digital Flight Control and Avionics Systems	5-6
5.2.1 Digital Data Buses	5-7
5.3 Software	5-11
5.3.1 Coordinate Systems	5-11
5.3.2 Integrated Control Core Software Modules	5-14
5.4 System Monitoring and Error Management	5-14
5.4.1 Processor Failure Detection	5-14
5.4.2 Data Transmission Error Detection	5-16
5.4.3 Data Validity	5-17
5.5 Systems Configuration	5-18
5.5.1 Simplex Configuration	5-19
5.5.2 Single-String Configurations	5-19
5.5.3 Dual System Configuration	5-20
5.5.4 Dual-Dual System Configuration	5-23
5.5.5 Triplex	5-25
5.6 ARINC Subsystems	5-27
5.6.1 Area Navigation System	5-27
5.6.2 Flight Control Computer System	5-31
5.6.3 Flight Management Computer System	5-39
5.6.4 Thrust Control Computer System	5-39
5.6.5 Inertial Reference System (IRS)	5-45
5.6.6 Attitude and Heading Reference System	5-45
5.6.7 Air Data System	5-51
5.6.8 Radio Altimeter	5-51
5.6.9 Airborne Weather Radar	5-51
5.6.10 Airborne Distance Measuring Equipment	5-55
5.6.11 ILS Receiver	5-55
5.6.12 Airborne VOR Receiver	5-56
5.6.13 Airborne ADF System	5-56
5.6.14 Mark 2 Omega Navigation System	5-56
5.6.15 Airborne VHF Communication Transceiver	5-56

TABLE OF CONTENTS (Continued)

	Page
5.6.16 Flight Data Acquisition and Recording System	5-57
5.6.17 Mark 3 Air Traffic Control Transponder (ATCRBS/DABS)	5-57
5.6.18 Digital Frequency/Function Selection for Airborne Electronic Equipment	5-57
5.6.19 Ground Proximity Warning System	5-57
5.6.20 Electronic Flight Instruments (EFI)	5-59
5.6.21 Flight Warning Computer System	5-60
5.6.22 Airborne MLS Receiver	5-60
5.6.23 Analog and Discrete Data Converter System	5-60
5.6.24 Airborne Separation Assurance System	5-60
5.6.25 Electric Chronometer	5-61
5.6.26 Digital Engine Controller	5-61
5.6.27 Electric Power Generation System	5-61
5.6.28 Synopsis of Digital Interfaces of ARINC Characteristic Subsystems	5-62
 5.7 References	 5-67

LIST OF ILLUSTRATIONS

Figure		Page
5-1	Fault Tolerance Definitional Relationships	5-2
5-2	Fault Types	5-2
5-3	ARINC 429 General Word Format	5-7
5-4	MIL-STD-1553B Word Format	5-10
5-5	MIL-STD-1553B Information Transfer Formats	5-10
5-6	Inertial Coordinate System	5-12
5-7	Earth Fixed Coordinate System	5-12
5-8	Body Axes Coordinate System	5-13
5-9	Locally Level Coordinate System	5-13
5-10	Horizontal Plane Coordinate System	5-15
5-11	Flight Management Computer System Configuration 2: Single System/Dual CDU Installation	5-20
5-12	Flight Management Computer System Configuration 3: Dual System Installation	5-21

LIST OF ILLUSTRATIONS (Continued)

Figure		Page
5-13	Dual Digital Flight Control System With Inter-Unit Switching (Ref. 39)	5-22
5-14	Typical Dual-Dual AFS Configuration	5-24
5-15	Typical Triplex DAFS Architecture	5-26
5-16	ARINC 701 Interfaces	5-32
5-17	Flight Management Computer System Interface With Flight Control System and Other Subsystems	5-40
5-18	ARINC 703 Thrust Control Computer Interface	5-44
5-19	704 Block Diagram	5-49

LIST OF TABLES

Table		Page
5-1	Dits Buses	5-8
5-2	ARINC Characteristic 581 Functions	5-28
5-3	ARINC 581 System Inputs - Range and Resolution	5-28
5-4	ARINC Characteristic 581 System Operation With Failed Sensors	5-29
5-5	ARINC Characteristic 583 Basic Functions	5-30
5-6	ARINC 583 System Inputs Range and Resolution	5-30
5-7	Example of ARINC Characteristic 583 System Operation With Failed Sensors	5-31
5-8	ARNIC 701 Flight Control Computer (FCC) System Controller Functions	5-36
5-9	FCCS Data Words	5-37
5-10	FCCS Discrete Input and Output Word Formats (Discrete Word #1)	5-37
5-11	FCCS Discrete Word #2	5-38

LIST OF TABLES (Continued)

Table		Page
5-12	ARINC 702 Flight Management System Functions	5-41
5-13	Thrust Control Computer System Components and Functions	5-43
5-14	IRS Digital Summary - Inputs/Outputs (Reference 51)	5-45
5-15	Digital Control Word Format IRS Discrete Word Format	5-47
5-16	AHRS Digital Output Summary	5-50
5-17	Air Data System Digital Data Output Standards (Reference 53)	5-52
5-18	ADS Discrete Word #1 Format (Reference 53)	5-53
5-19	ADS Discrete Word #2 Format (Reference 53)	5-54
5-20	709 DME Mode Selection Matrix	5-55
5-21	Discrete Work Format (Reference 66)	5-58
5-22	ARINC 725-1 Signal Generator Digital Inputs (Reference 67)	5-59
5-23	ARINC 726-1 Flight Warning Computer System Digital Inputs (Reference 68)	5-60
5-24	ARINC 730-3 Airborne Separation Assurance System Digital Inputs (Reference 71)	5-61
5-25	Synopsis of Digital Interfaces of ARINC Characteristic Subsystems	5-63

SECTION 5

5. SYSTEM ARCHITECTURES

5.1 FAULT TOLERANT DIGITAL INTEGRATED FLIGHT CONTROL AND AVIONICS.

The requirements for advanced digital integrated flight control and avionics systems are based upon the performance, reliability, and safety factors presented in Section 4. In addition, operational environment factors, such as lightning, impact the ability of a system configuration to meet the reliability and safety requirements. A system is normally designed to tolerate a specific set of faults. A system designed to tolerate all known fault types, however small the probability that a fault of a specific type might occur, could be prohibitively expensive.

A fault tolerant system may be considered to be a system which provides the correct execution of a function at all times and encompasses:

- (1) Elimination of hardware design errors.
- (2) Correctness and completeness of software specification.
- (3) Testing, verification, and validation of programs and microprograms.
- (4) Continued correct execution of programs in the presence of hardware (physical) faults.

Figure 5-1 illustrates the basic relationship which must be clearly understood in order to avoid the confusion of the terms errors/fault/failures. These definitions should be the same for hardware or physical devices as well as software. Unfortunately, different authors have used different definitions (references 1 through 5). This has resulted in communication problems and use of the phrase "software error" when the correct phrase should have been "software failure." A failure of hardware or software corresponds to the cause. An unspecified and disruptive change in one or more logic variables of the digital system is a fault. "The occurrence of a fault often causes an error; that is, a deviation of the logic machine from its programmed - specified behavior (transition through a sequence of specified stage) into a sequence of error states" (reference 5). To summarize, a failure is the cause and the error is an effect.

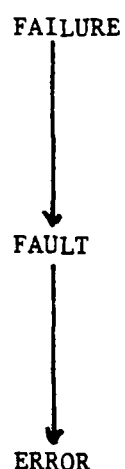
Figure 5-2 depicts the two major classifications of fault types. These are the physical (operational) faults which are those faults which occur while the fault tolerant integrated control system is operating (both in flight and on the ground), and the manmade faults.

"A physical fault is caused by physical failure phenomena that could affect one or more components of the system and causes either a permanent or temporary change to the values of the physical variable" (reference 5). The time duration of a physical fault determines whether the fault is a permanent or transient fault. Transient faults are of limited duration caused either by a temporary malfunction of components or by external interference. Each transient fault, or combination of transient faults, must be translated into an allowable failure reaction time in order to select suitable recovery techniques. Transient fault durations, which

Relational Sequence



Fault Tolerant Parameters



"omission of occurrence or performance; specifically: a failing to perform a duty or expected action"*. (Websters Seventh New Collegiate Dictionary.)

"Variation in measurements, calculations, or observations of a quantity due to mistakes or uncontrollable factors"*

FIGURE 5-1. FAULT TOLERANCE DEFINITIONAL RELATIONSHIPS

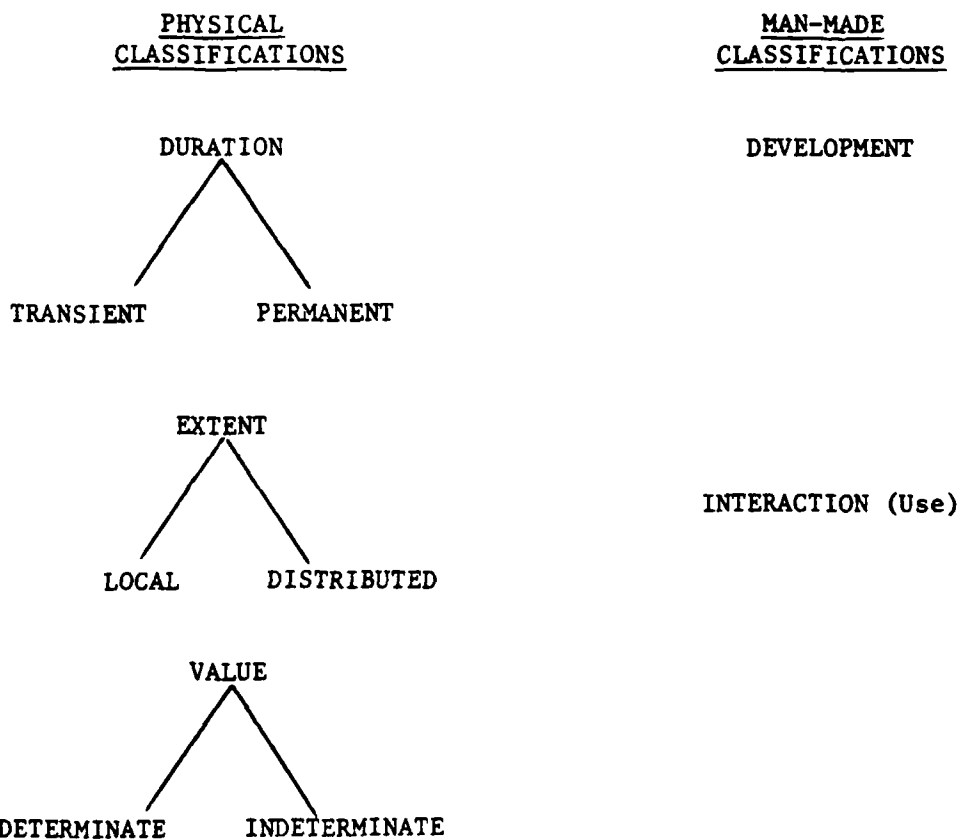


FIGURE 5-2. FAULT TYPES

exceed the failure reaction time would be considered permanent faults. Permanent faults are caused by complete failures of components for which recovery can be accomplished either by the presence of a duplicate (spare) component or by provisions to continue operation without the failed components.

Distributed and local faults differ as to the extent of the number of parameters simultaneously affected by single failure. Local (single) faults affect only single logic variables while distributed (related multiple) faults are those that affect two or more variables, one module, or an entire system. For example, an electrical power bus failure may affect many integrated control subsystems, whereas, a power supply failure within a single subsystem may affect only that subsystem. Distributed faults are much more likely in medium- and large-scale integrated circuitry than in discrete component designs due to the physical proximity of the logic element on the chip. Distributed failures are also caused by single failures of a critical element in a computer system (e.g., clock, power supply, data bus, switches for reconfiguration, etc.).

The value of a fault may be considered to be determinate or indeterminate. Determinate faults result in a logic value assuming a constant (stuck at "one" or "zero") value throughout its entire duration, whereas, an indeterminate fault results in variation in the logic value but not in accordance with the design specification.

"Manmade faults are defined for the purpose of this report to fall into two major classifications. These are development faults and interaction or use faults. The development faults can occur at any time during the development phase including the original requirements definition. As a result, incomplete, ambiguous, or erroneous specifications may be used as the basis for the development of the design. Even if a specification is totally correct, 'mistakes' may occur during the various phases of translation of a correct specification into the final developed system. These 'mistakes' may involve hardware or software design, as well as imperfections in 'programming and maintenance tools such as compilers, assemblers, design automation programs, maintenance and operation manuals, testing procedures and devices, etc.'" (reference 5).

Interaction (use) faults have been defined as "faults caused by inputs that are introduced into the system via the man-machine interfaces during operation or maintenance phases by the operator, and which are not appropriate to the current state of system. Inappropriate inputs are caused typically either by misunderstanding of the operator's manual or by typographical errors that occur while information is being entered into the system" (reference 5).

Two complementary approaches have been used to control physical and manmade faults. These approaches are:

- (1) Fault avoidance in which the resources are allocated to obtain a high probability of correct operation by use of high reliability parts with no redundancy. In this approach, manual maintenance is required to return the system to an operating condition whenever failure occurs. This approach is costly, not only during the acquisition phase but also during the operation phase, due to the requirement of having highly skilled maintenance personnel available.

(2) Fault tolerance which utilizes a protective redundancy approach in which both hardware and software elements of the system may be redundant and can be either active or standby.

The fault tolerant approach allows the designer to trade off reliability, availability, and cost. The designer must have knowledge of the types of operational faults and their impact on the integrated control system functions, redundancy techniques which can be implemented to protect against these faults, and the design tools required to effect the different redundancy mechanizations.

5.1.1 Redundancy Techniques.

5.1.1.1 Hardware. Static hardware redundancy utilizes redundant copies of components permanently connected and powered. These components provide fault masking either through approaches such as triple modular redundancy with voting or replication of the individual electronic component. It is assumed that failures are statistically independent and that the redundant elements have the same failure rates and are instantaneously available to perform the masking of the failure with unity probability of success. The reliability of a static redundant system is obtained as the sum of reliabilities of all distinct configurations (including none or some failed parts) that do not lead to system failure.

Dynamic hardware redundancy requires automatic fault protection and subsequent recovery by either eliminating the fault or correcting the error created by the fault. If static reliability models are used for the dynamic case, the assumption is often made that the probability of detecting the fault and recovery is unity. Since this is known to be impossible, using state-of-the-art built-in test methods, most models have added the concept of coverage which is defined as conditional probabilities of successful recovery given that a fault has occurred. This permits treatment of the case in which a fault is not detected, as well as the case in which a fault is detected and a correction is not made. Dynamic redundancy models must represent the entire complexity of the proposed fault tolerant system including:

- (1) Differing failure rates for powered and unpowered modules.
- (2) Increases in module failure rates due to the fault detection and switching hardware.
- (3) The number of spares of each module.
- (4) Imperfect fault detection and system recovery.
- (5) Extent and value of the fault.
- (6) Duration and distribution of the expected transient fault.

5.1.1.2 Software Redundancy. Software redundancy is usually employed in conjunction with dynamic hardware redundancy. Major forms of software redundancy are:

- (1) Multiple storage of critical programs and data.
- (2) Test and diagnostic programs or microprograms.
- (3) Parts of the operating system which interact with hardware redundancy in carrying out program restarts.

5.1.1.3 Time Redundancy. Timing or execution redundancy is usually employed together with dynamic hardware and software techniques. Techniques which have been used include repeated execution or acknowledgement of operation and recovery by program restart or operational retries after fault detection or reconfiguration has occurred. Time redundancy has not been used extensively in present operational avionic systems. The Digital Avionics Information System (DAIS) employs a limited form of time redundancy in which fault detection is implemented by repeated execution or acknowledgement (handshaking). (Retransmission of the message is a typical error correction method in software redundancy.) The DAIS also utilizes the time redundancy in the identification and correction of errors caused by transient faults and in program restarts after a hardware reconfiguration. Great care must be taken by the programmer when time redundancy is used since singular events should not be repeated.

5.1.2 Motivation for Fault Tolerant Integrated Control.

Recent advances in airborne computational capability, and electronic reliability, have provided aircraft designers with the opportunity to develop integrated control system designs to achieve greater vehicle performance and operational efficiency, as well as longer aircraft life. Active control systems are being investigated and developed to provide flutter control and load alleviation and to allow reduced aerodynamic static stability. Substitution of electronic controls for more conventional structural and aerodynamic safety features implies a need for an inherently reliable "flight critical" control system architecture. The conventional technique for achieving the necessary confidence in the control system has been to provide parallel redundant systems with appropriate voting and monitoring in order to detect and remove failed systems. The number of redundant elements which can be used is limited by the complexity of the voting and monitoring functions. Reliability and safety analyses have shown the traditional approach is inappropriate for a system which is critical throughout all flight phases. Motivation for the development of specific fault tolerant integrated control systems (for future aircraft) will be based ultimately upon life cycle cost considerations. Fault tolerance appears to be the most feasible way to achieve the high reliability levels essential to the deployment of economically competitive system functions. Fault tolerant system design can also be shown to minimize the risk of loss of life or heavy damage to the aircraft and thus may provide psychological support to the users (e.g., the system is ready for faults) (reference 6).

Fault tolerant flight control system design alternatives have been studied extensively (references 7-18). A number of these studies have been concerned with the architecture (references 2, 3, 5, and 18) and specific subsystems such as sensors,

computers, actuators, and software. The various fault-tolerant mechanizations all require redundancy in the sensors, computers, and actuators. Approaches taken in the sensors area include: (a) in-line redundant sensors for each of the redundant channels; (b) skewed and special sensors; (c) integrating sensor data, from subsystems normally not functionally related, for monitoring and tie breaking; (d) in-line monitoring (self-test) sensors; and (e) analytical redundancy. The analytical redundancy concepts range from simple signal blenders to complex Kalman filters (references 7 through 9 and 19 through 24). In addition, there are various voting configurations that include sensor voting using cross-channel links as well as sensor data cross strapping with voting at each of the intermediate levels, such as input to the computers, and finally input into the actuators.

The computer element in the fault tolerant system ranges from complete computer redundancy to redundancy of the elements of the computer, such as the microprocessor or central processing unit, input/output interfaces, memory, clocks, and power supplies (references 13, 25, and 26).

To summarize, the fault tolerant integrated control systems will have high availability, capability which can be maintained or degrades gracefully with failures, and life cycle cost benefits.

5.2 ADVANCED INTEGRATED DIGITAL FLIGHT CONTROL AND AVIONICS SYSTEMS.

The basic elements of advanced digital integrated flight control and avionics systems include the hardware (sensors, data buses, processors, displays, and actuators), software, and the crewmembers. The architectural variations possible for the implementation of the design involve primarily hardware and software variations. The crew architectural variations are primarily limited to the partitioning of functions between the system and the crewmembers.

Hardware includes the sensors, actuators, displays, computers, digital information transfer system (ARINC 429, ARINC 453, or MIL-STD-1553A/B and others to be determined) electrical power, and hydraulic systems. The sensors may, in themselves, be redundant with their outputs voted, or the sensors may, while not identical, be redundant in an analytical manner in which the software implemented in the computers compares the outputs by employing the known physical relationships between the sensors.

Actuators effect the desired control through the control surface. In addition to the possibility of actuator redundancy, the control surfaces may be segmented to achieve control redundancy.

The display device may integrate and present the information required by the crew with the priority of information determined by the flight phase and crew selection or the devices may be redundant with dedicated information.

The computers may be functionally redundant or they may be physically redundant. Functional redundancy may involve sharing elements through the internal bus architecture of the computer. In addition, the computing subsystem may have multiple central processing units (CPU) sharing common (global) memory using one of the many bus control protocol schemes possible with today's technology and microarchitectures. A network of computers, such as that becoming commonplace in avionics systems, requires a digital communication standard in order to permit efficient exchange of data between and among the computers; and allow for the integration of

any other computer which complies with the interface requirements of the digital communication standard. The physical implementation of the standard is done through a digital data bus which is composed of wire bundles, shielded cables, and/or fiber optic bundles. The following section synthesizes the characteristics of some of the data buses which are currently being utilized or being considered for aircraft avionics and flight control systems integration.

5.2.1 Digital Data Buses.

Table 5-1 lists seven buses and their various characteristics. The last four buses in the table are listed because various government agencies and manufacturers have been experimenting with these buses, such as the IEEE 488-1978, in actual flight tests, or the buses are the more common microprocessor buses used for internal connection of components of the microprocessors.

5.2.1.1 ARINC 429 (MARK 33). The ARINC 429 (reference 27) is a serial, unidirectional 2-wire (shielded twisted pair) bus. Data are sent in broadcast form with a terminal sending information over the bus in only one direction. All return information must be placed on a separate bus. Data may be sent in either binary or alpha/numeric form. An 8-bit label is present in each 32-bit word to identify the type of information being sent as shown in figure 5-3. Label 377 is reserved to indicate transmission of the equipment identification. There are 19 data bits or 3 alpha/numeric characters per word. Bit numbers 30 and 31 are the sign/status matrix indicator. Bit numbers 9 and 10 of numeric data words provide 3 device addresses per bus. The least significant bit (LSB) of the word (bit Position 1) is transmitted first. Bit 32, parity, is transmitted last. All information update rates are specified in this standard. The high data rate is 100 KHz which comes to less than 2800 words per second with 4-bit times between words. The low speed operation shall have a bit rate within the range of 12-14.5 kilobits per second. The bus has been found to operate at lengths of over 200 feet.

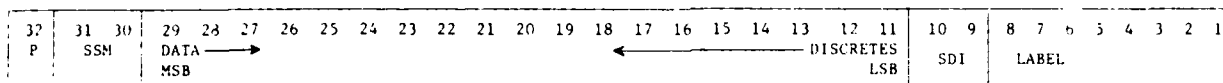


FIGURE 5-3. ARINC 429 GENERAL WORD FORMAT

5.2.1.2 ARINC 453. The ARINC 453 (reference 28) is a serial, 2-wire (shielded twisted pair), broadcast bus. It is also unidirectional, but provisions for multiple transmitters imply bidirectional. The electrical characteristics are similar to 1553A. The data format is the same as MARK 33, except for the addition of a synch waveform and an extended alpha/numeric handling format. In two transmitter systems, a transmitter may take control of the bus only when it senses no signal on it. In other systems, the control priority can be determined by delaying takeover of the bus by varying amounts after an empty bus occurs. Another method is through the use of a bus controller, whose characteristics are not described in the standard.

5.2.1.3 MIL-STD-1553B. The MIL-STD-1553B bus (reference 29) is a serial bidirectional, 2-wire (shielded twisted pair) bus. Data are transmitted in a command/response protocol with the bus controller sending the transmit and receive commands to the appropriate devices. A status word is transmitted by the remote

TABLE 5-1. DITS BUSES

SERIAL/ PARALLEL	SIGNAL LINES	WORD SIZE BITS	DATA BITS	BUS PROTOCOL DIR-CTIONAL	GAP	SPEED/CLK	ADDR/LINES	INTRPTS	NOMINAL VOLTAGE LEVEL	MODULATION CODE	MAX LINE LEN	
MIL-STD-1553B	S	2	20	16	NA	B1-	C/R*** 1MHz	32	NA	+3 to +10	2 level Manchester	300
DRAFT ARINC 453	S	2	36	14	NA	B1-	BR* 1MHz	20 receivers 3 addresses	NA	+3 to +10	2 level Manchester	300
ARINC 429 MARK 33	S	2	32	19	4 ^a	UNI-	BR 100KHz ^b 12-14.5KHz	20 receivers 3 addresses	NA	+10v	Bipolar Return to zero 3 level	
IEEE 488-1978	P	16	8	8	NA	B1-	C/R 1MHz*	15 receivers 1024 addresses	POLLING	TTL	TTL	66
DEC UNIBUS	P	56	16	16	NA	B1-	C/R 2.5MHz max	218	5**	TTL	TTL	50
INTEL MULTIBUS IEEE 796	P	55	16	16	NA	B1-	C/R 5MHz	216	8	TTL	TTL	
S-100 IEEE 696	P	100	8/16	8/16	NA	B1-	C/R 2-4MHz	216/224	8	TTL	TTL	180

*Depends on number of terminals and bus physical length

**Depends on level and bus position

***Command/Response

-Broadcast

^aBIT TIMES WORD GAP
^bHigh-Bit-Rate and Low-Bit-Rate Messages
will not be intermixed on the same bus.

terminal whose address is in the command word. The command words can address 32 separate terminals with up to 31 subaddresses or modes in each. Data are transmitted in blocks of up to 32 words with the most significant bit (bit time 1) of the first word transmitted first with the less significant bits following in descending order. These blocks are transmitted at 1 MHz with word length being 20 bits, 16 of which are data as shown in figure 5-4; figure 5-5 gives the different information transfer formats.

The maximum length is unspecified for 1553B. Data broadcast is not allowed in the USAF applications of the bus (reference 30).

5.2.1.4 IEEE 488-1978. The IEEE 488-1978 bus (reference 31) comprises an 8-bit parallel data bus, 3 data-byte transfer control lines, and a general interface management bus consisting of five signal lines. Message bits are carried on the eight parallel line in a bit-parallel, byte-serial form, asynchronously, and generally in a bidirectional manner. Since a device can be able to transmit, receive, and control; or to transmit and receive only; or to receive only; or to transmit only; the generally bidirectional nature of the data bus becomes apparent. At the same time, it is difficult to characterize the communication protocol as other than a command/response system. The transmission rate on the bus is, therefore, limited to that of the slowest device on the bus involved in the handshaking which constitutes the protocol. Other limitations include the limit of 15 addresses on the bus, and the bus length between terminals cannot be more than 2 meters while the total transmission path length over the interconnecting cable cannot exceed 20 meters.

5.2.1.5 DEC UNIBUS™. The Digital Equipment Corporation UNIBUS (reference 32) is a 56-line bidirectional bus operating in a master/slave relationship. There are 16 data lines with the maximum transfer rate of 2.5 MHz. Maximum bus length without repeaters is approximately 50 feet.

5.2.1.6 Intel MULTIBUS™. The Intel Multibus/IEEE-796 (reference 33) supports multiple processor systems with its multimaster bus structure. Each processor has its own local bus, memory, input/output, and has access to the multiple master system bus (Multibus) through Multibus control and buffers contained on the same board as the processor. Access to the system bus is requested only when a common memory or input/output, resident on the bus and accessible by the multiple masters, is referenced during an instruction execution cycle. If the address of the memory device or input/output location is a global address outside the range of the onboard memory or input/output, a system bus request is initiated. Since it is possible that another master may be currently utilizing the system bus, an arbitration procedure must be provided to resolve the multiple demand problem for this system bus. The Multibus structure provides two arbitration techniques: (1) serial in which the priority is ordered on the basis of bus location or (2) parallel in which hardware encoding is used to resolve system bus master priorities. The Multibus is bidirectional with a 16-bit data line and a 16-bit address line. The bus may be considered to be a command/response type bus with the transmission rate dependent upon the speed of the processor (the MULTIBUS supports the 8080, 8085, and Z80 and other microprocessors). Maximum bus length is approximately 50 feet without repeaters.

5.2.1.7 S-100 (IEEE 696) Bus. The S-100 bus (reference 34) was originally designed by MITS, Incorporated, and has become a standard to the point that the IEEE has, with some modification, adopted the S-100 as a standard bus (IEEE 696).

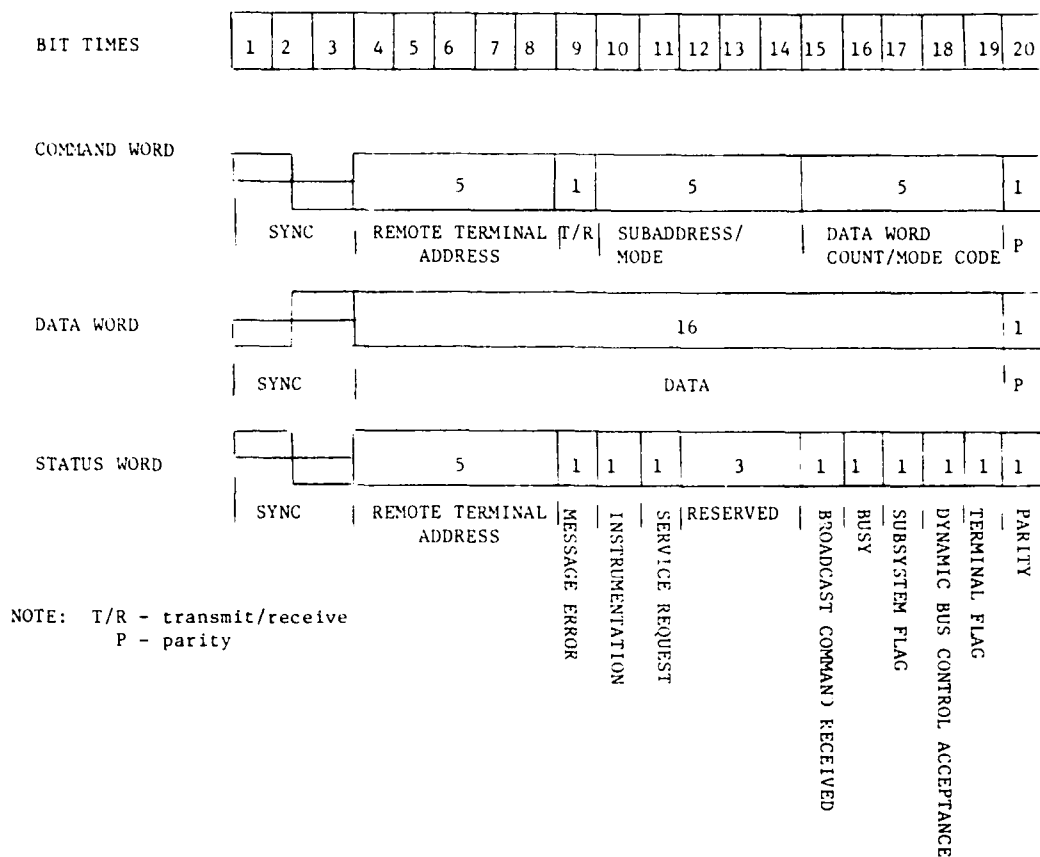
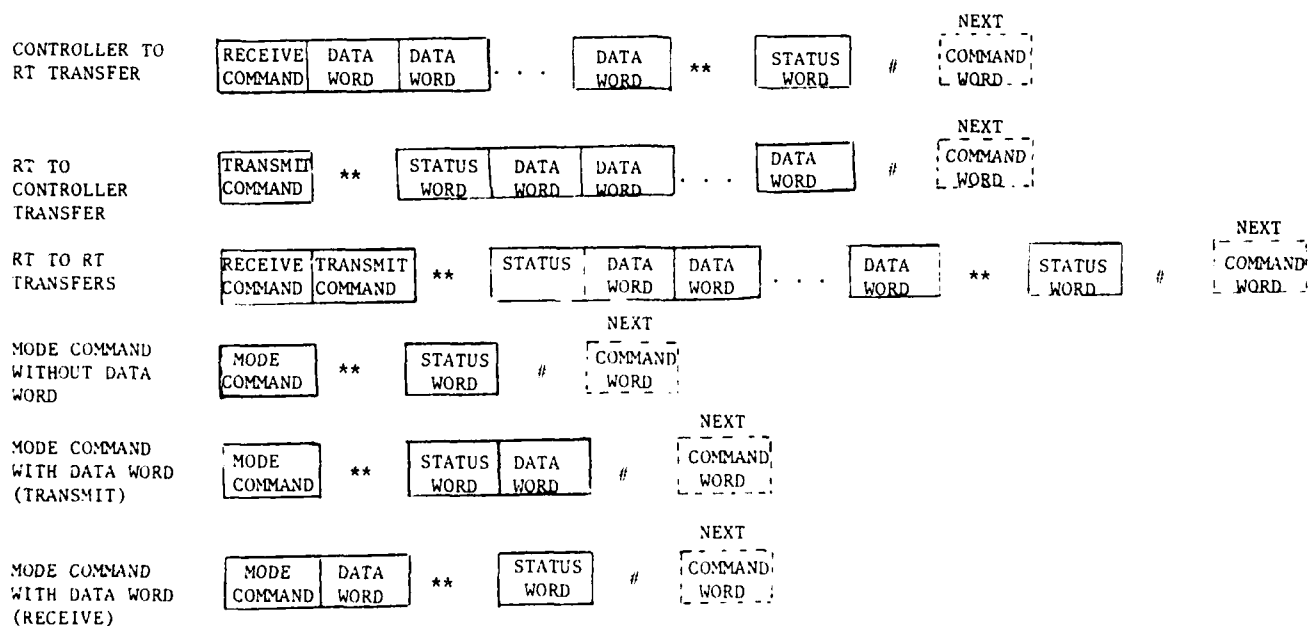


FIGURE 5-4. MIL-STD-1553B WORD FORMAT



NOTE: # INTERMESSAGE GAP
** RESPONSE TIME

FIGURE 5-5. MIL-STD-1553B INFORMATION TRANSFER FORMATS

This bus uses 16 address lines and separate 8-bit parallel data output and data input buses. The bus protocol was adapted from the bus master/bus slave concept of Digital Equipment Corporation's PDP-11. Extensions to the S-100 bus permit 16-bit parallel data and extended addressing (up to 24 address lines). For 16-bit masters and slaves, the data input and data output lines form a 16-bit bidirectional data bus with the data out lines carrying the lower order byte while the data in lines accommodate the higher order byte. This is accomplished by asserting a 16-bit request status signal followed by asserting the 16-bit response signal. If the 16-bit option is used, the word length increases from 8 bits to 16 bits.

5.3 SOFTWARE.

The architecture of the software implemented in digital flight control and avionics systems must be known. The structure and relationship of the modules, input/output parameters, and coordinate systems are vital to the correct operation of the system.

5.3.1 Coordinate Systems.

Validation of aircraft software requires complete knowledge of the frames of reference within which the various sets of equations are cast. Five of the commonly used coordinate systems are the earth fixed, inertial, body axes, local level, and horizontal plane frames of reference. Although these systems are often similar in intent to the ones actually being used, the descriptions differ somewhat in terminology and in their definitions of the various frames of reference. These differences tend to mask the underlying similarities between the software functions being performed. From an avionics software viewpoint, it is necessary to have a common and consistent set of definitions and terms (reference 35). In this handbook subsection, the various aircraft coordinate systems are described as a set of recommendations for standard frames of reference for avionics software.

5.3.1.1 Inertial Coordinate System. Figure 5-6 shows the axes for an inertial coordinate system. Common practice is to have two axes in the equatorial plane and the other axis coincident with the earth's angular velocity vector. The axes are nonrotating relative to the stars and have their origin at the center of mass of the earth (reference 35).

5.3.1.2 Earth Fixed Coordinate System. Figure 5-7 depicts a typical earth fixed reference frame. The earth fixed coordinate system is the same as the inertial coordinate system except in the fact that the X and Y axes are fixed to the earth and thus rotate. At time $t = 0$ (the navigation starting time), the two systems are identical (reference 35).

5.3.1.3 Body Axes Reference Frame. A body axes coordinate system is shown in figure 5-8. The X, Y, and Z axes represent vehicle axes of roll, pitch, and yaw, respectively. Positive directions are assumed to be forward, out the right wing, and down. Positive roll, pitch, and yaw follow the righthand rule. The coordinate system's origin is at the aircraft's center of gravity (reference 35).

5.3.1.4 Locally Level Coordinate System. The locally level reference frame is a local navigational frame which has its origin at the system's location and its axes aligned with the north (X), east (Y), and down (Z), where down is in the direction opposite to the earth radius. The locally level coordinate system is illustrated in figure 5-9 (reference 35).

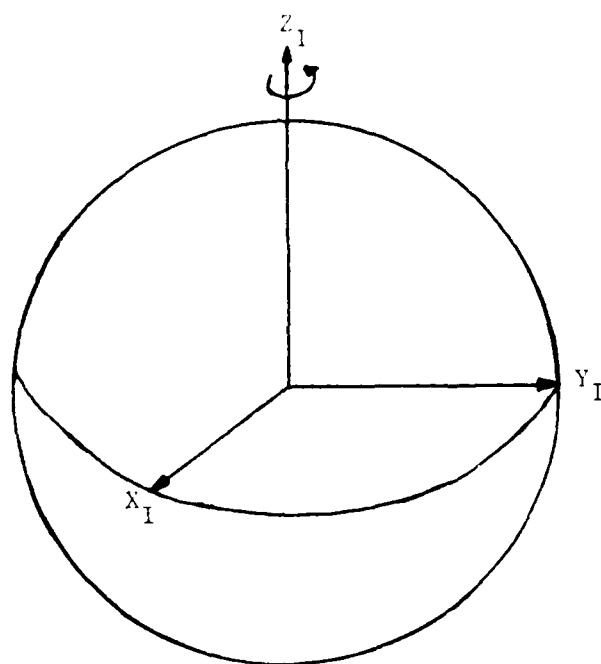


FIGURE 5-6. INERTIAL COORDINATE SYSTEM

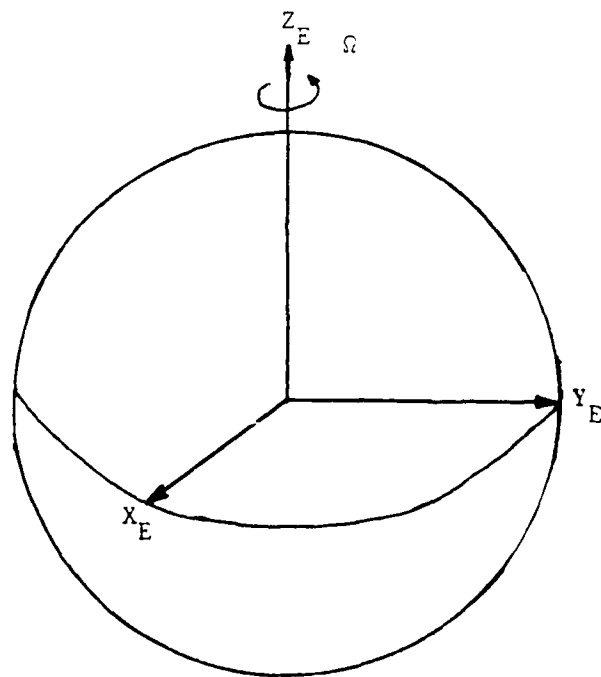


FIGURE 5-7. EARTH FIXED COORDINATE SYSTEM

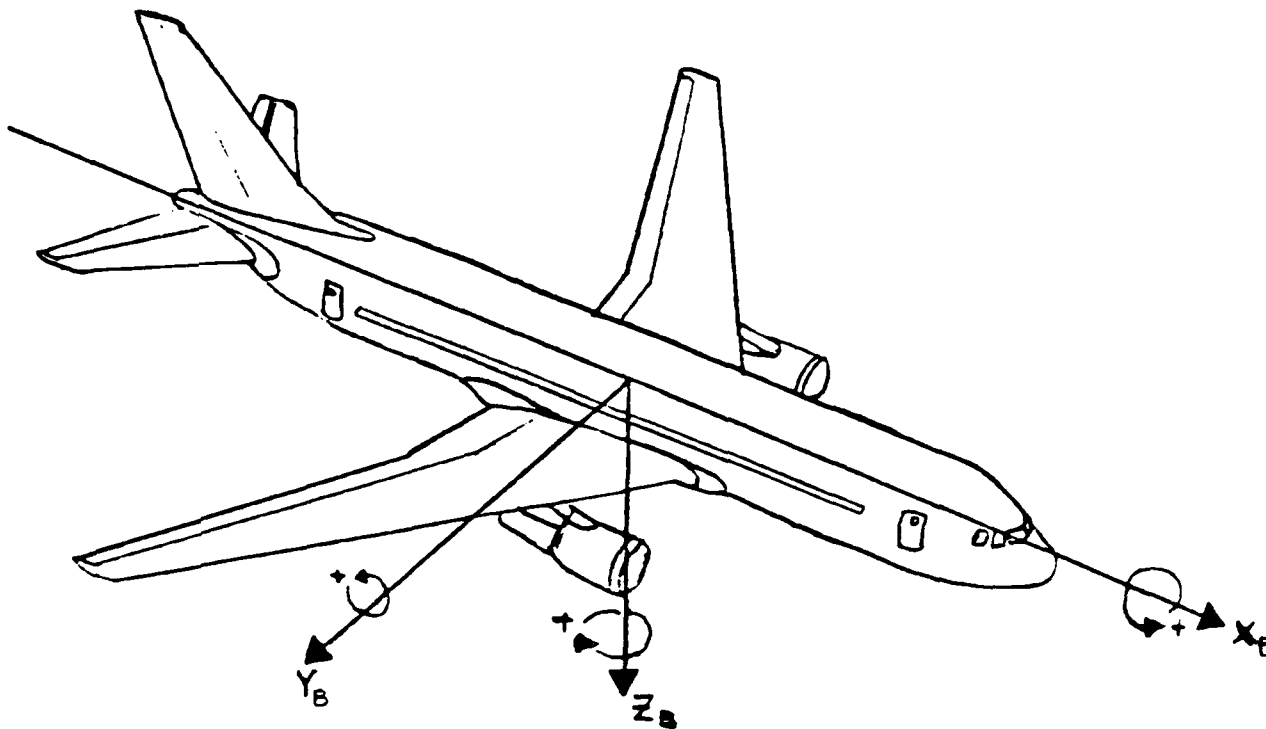


FIGURE 5-8. BODY AXES COORDINATE SYSTEM

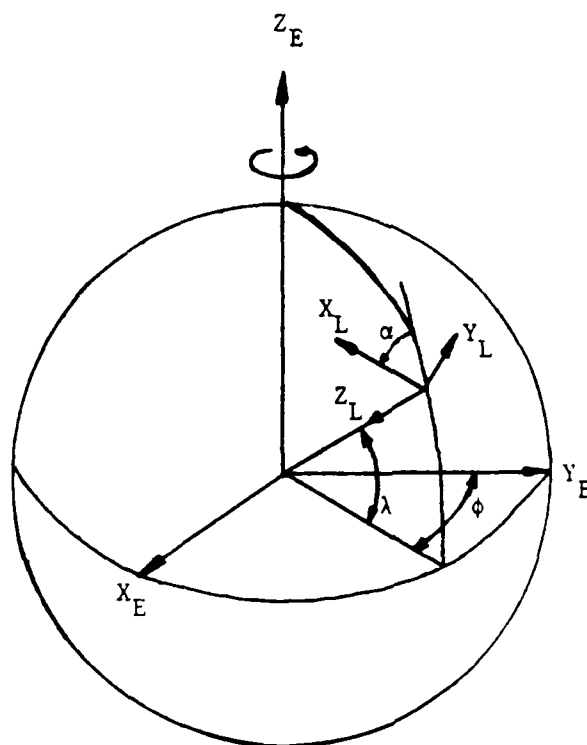


FIGURE 5-9. LOCALLY LEVEL COORDINATE SYSTEM

5.3.1.5 Horizontal Plane Coordinate System. Figure 5-10 depicts a horizontal plane reference frame and the associated parameters. Measurements taken from true north are positive in a clockwise direction. The horizontal plane coordinate system is often referred to as the navigation reference frame (reference 35).

5.3.2 Integrated Control Core Software Modules.

Integrated control systems employing digital technology require an executive for control of communication protocol when a multiplex data bus system is used, as well as a local executive within each processor. The executive program provides the real-time control of the application software which comprises the tasks that collectively perform specific integrated control of flight tasks. The executive implements, in general, a scheduling mechanism, input/output (I/O), error handling, and startup and initialization. The error handling algorithms include the requirement for error detection including transient failures, and error correction which may involve backup/ recovery or reconfiguration of the integrated control system.

The application software includes state estimation, guidance, and control functions. The estimated state vector may be derived using redundant inputs from various state measurement sensors or may involve analytic redundancy (reference 9) or combinations of both (reference 8).

In the case of manual control, the need exists to not only determine the information required by the crew but also the manner in which it should be presented to the crew.

5.4 SYSTEM MONITORING AND ERROR MANAGEMENT.

Digital avionics systems should make use of an array of system monitors in both hardware and software. These system monitors are designed to detect errors in the digital system's hardware and software execution, and either correct these detected errors or set a flag indicating the error type. The error management techniques are dependent upon the system architecture and may be quite different for single string, dual, and other forms of redundancy such as dual-dual and triplex.

As a minimum, the error detection must consider three levels. These three levels should be implemented to assure:

- (1) Correct operation of each processing unit.
- (2) Valid transmission of data between digital subsystems.
- (3) Data validity prior to use in subsequent computation.

Each of these levels is discussed in the following paragraphs. The examples given are illustrative of those being used in digital systems and are not meant to indicate one method is preferred over another. The principal concern is to make sure that a method which works is employed at each level.

5.4.1 Processor Failure Detection.

Two conventional methods for detecting failure of the processing units are self-test and comparison monitoring. In self-test, each component of the processor is exercised by a set of computations designed specifically to test that component (references 36 and 37). The results of each computational set are compared with prestored values and any differences signify that a fault was detected. Typical computations used include:

- (1) Memory Parity Check
- (2) Checksum
- (3) Read-Write Memory Functional Check
- (4) CPU Instruction Set Execution Tests
- (5) Task Completion Flags (Computer Cycle Time Test)
- (6) Foreground/Background Checks
- (7) Watchdog Monitor Pulse (reference 36)
- (8) Automatic Fault Injection (reference 36)

Automatic fault injection can be designed to detect a high percentage of the following types of faults:

- (1) Integrated circuit devices -- output stuck at 0 or 1, output open, or input open.
- (2) PROM/ROM pin faults -- same as (1) above.
- (3) PROM/ROM bit faults -- state change.
- (4) Connector pins -- open.
- (5) Discrete components -- open and short.
- (6) Multilayer boards -- open runs and solder joints.

Tests have been conducted using fault injection experiments to determine the time-to-detect a fault by comparison monitoring (reference 37). Detection is assumed to occur whenever there is a difference between the computed outputs of the faulted and non-faulted processors executing the same program. The results of this test correlated with previous observations "that comparison-monitoring yields a detection coverage which ranges from 40 to 60 percent and is in sharp contrast to assumed values of unity for first failure coverage in comparison-monitoring or majority-voting (cm/mv) systems" (reference 37).

5.4.2 Data Transmission Error Detection.

Data transmission validity checks should include at least one or more of the following:

- (1) Parity (ARINC 429 and MIL-STD-1553B require odd parity for valid transmission).
- (2) Word length count.
- (3) Message length count for MIL-STD-1553B.
- (4) Address Validity for MIL-STD-1553B.
- (5) Modulation Waveform Test.
- (6) Intermessage Gap Time Test.
- (7) Minimum No-Response Time-Out for MIL-STD-1553B.
- (8) Contiguous Message Transmission for MIL-STD-1553B.
- (9) Dead Line Time-Out.

While ARINC 429 makes use of some of the encoding methods used in digital network data transfer, it does not implement a complete set of error detection/correction codes such as that often used in digital data transmission. The six encoding methods used in ARINC 429 (reference 27) are:

- (1) BNR (two's complement fractional binary notation)
- (2) BCD (Binary coded decimal notation) per the numerical subset of ISO Alphabet No. 5

- (3) Discrete data
- (4) Maintenance data (general)
- (5) Acknowledgement, International Standards Organization (ISO) Alphabet No. 5, and Maintenance data (AIM)
- (6) File Data transfer

ISO Alphabet No. 5 is a character-oriented protocol. While character-oriented protocols can make use of Vertical Redundancy Check (VRC), Longitudinal Redundancy Check (LRC), and Cyclic Redundancy Check (CRC), ARINC 429 makes use of only the VRC in which the 32nd bit is calculated to provide an "odd" number of bits in the word. The VRC scheme can only detect an odd number of bit errors in the transmission. In the file data transfer, a file may contain from 1 to 127 records with each record containing from 1 to 126 data words. In this transmission method, the "final word" in each record is an "error control" word. "Bit numbers 1 through 8 contain the file label. Bit numbers 9 through 29 contain an error control checksum computed from the states of bit numbers 9 through 29 of all intermediate words of the record. The error control checksum should be generated by the arithmetic addition of the binary values of bits 9 through 29 of all intermediate words and discarding the overflow" (reference 27). If the receiver detects an error by processing the error control information in the "final word," it should send a "Data Received Not OK" word to the transmitter.

Error management for data transmission errors which are detected is dependent upon the amount of redundancy in data buses in a single string system having no redundant subsystem, as well as the architecture/topology of the system having redundant subsystems. This subject will be discussed further in the following portion of this section which discusses system configurations.

5.4.3 Data Validity.

Even though there may be no error in the transmission of data between subsystems connected by the digital data buses, this does not mean that the data transmitted are valid. The systems engineer needs to consider other means of ascertaining that the data transmitted and received are valid. These checks should be designed to test the sensors inputs, conversion of sensor data to digital format, processing of these data, and conversion of the digital output data to the form required by the actuators or sensors using the result of the operation. These checks include (references 38, 39-41):

(1) Input range limit test--sensitive only to failure (open or short) which could produce a hardover signal.

(2) Input rate of change test--compares current value of input against previous value and, using time elapsed between data samples and physical laws, determines if input rate of change is reasonable.

(3) Parameter correlation check--compares redundant parameter data words to determine if their difference in value falls within/outside accuracy bounds. "Failure" in one of the two parameters is indicated when the accuracy bounds are exceeded for a given number of consecutive program cycles.

(4) Parameter majority logic check--this is a comparison of triple redundant parameter data words to determine if their differences in values fall within/outside accuracy bounds. Failure in one of the three parameters is indicated by

excessive error between its value and that of the other two good parameters for a given number of consecutive program cycles.

(5) Output wraparound test--torque motor/solenoid outputs are electrically fed back as inputs to the processor for a check by the software to detect output digital-to-analog (D/A) and torque motor/solenoid drive circuit failures and A/D input failures.

(6) Known input test--carried out under command of another computer.

It should produce a known output which is sampled by the input sensor. The known input and measured input resulting from feedback of the known input are compared. This is used for detection of a failed input sensor. Carried out in preflight only.

(7) Known output test--similar to the known input. Feedback of servo-position and rate of known output checks actuator operation.

(8) Loop dynamic check--control loop error (command value minus measured value) is compared against programmed limits. Failure is indicated when the measured error exceeds the programmed error for a given number of consecutive program cycles.

(9) End of conversion (EOC) bit not detected--after signaling start of data conversion to a digital data converter, an EOC bit should be set within an allowed conversion time. If the EOC bit is not set after the allowed time, a failed digital converter is indicated.

In addition to the above tests, power supplies should be monitored for intolerance operation. Critical sensors, such as the angle-of-attack sensors, which are duplicated, should be monitored. Multiple copies of critical data items should be compared, as a check to catch possible combinations of faults including sensor data conversion and data transmission. Hardware which includes (built in test equipment (BITE)) may output discretes which should be checked prior to use of the data.

Many additional types of tests exist to assure that the data have integrity. The foregoing list is not meant to be all inclusive and if a full description is provided with the hardware and software used in the system to be validated, the user of this handbook may recognize some of the tests being used as being variations of those in the foregoing list.

5.5 SYSTEMS CONFIGURATION.

The system configurations possible, using microcomputers and digital data buses for interconnection of the microcomputers, range from the simple single microcomputer with its own internal computer bus to configurations using 30 or more microcomputers and over 120 ARINC 429 data buses (e.g., the Boeing 767 (references 42-43)). This diversity in configurations results from the partitioning of system functions into more than one processor; and the fault tolerant design required for systems that are performing critical and essential functions (reference 44) in order to meet the respective extremely improbable or improbable requirements of reference 44. Critical functions are usually implemented as fail operational, which means that the function continues to operate in a normal manner after any single system component fails. Essential functions are usually implemented as fail passive

or fail soft, which means that a single failure will either cause loss of the function with no aircraft perturbation, or will cause a disturbance which is limited to a safe and acceptable level.

As discussed in reference 18, "Redundant copies, or channels as they are frequently called, can be configured as either independent channels or cross-strapped. 'Independent channels' indicates that there is no interconnection or sharing of control signals between the parallel channels. Cross-strapped means that there are interconnections and signal sharing between the redundant channels. Cross-strapping may be accomplished either by analog cross-feed or intercommunication between processors. Cross-strapping may be used at both the inputs (sensor-signals) and output (servo-drive) of the processors or at either point individually."

Redundant configurations can operate in either an active or active-standby mode. In the active mode, all redundant channels are operating simultaneously. In the active-standby mode, some of the channels are controlling the system while the others are standing by, ready to assume control in the event one or more of the active controlling elements is declared faulty by the fault detection logic. Some of the possible fault detection and system level error management concepts are discussed for each configuration of interest in the following paragraphs. These concepts are not the only possible error management concepts which have been considered by industry (references 7-10, 12, 18, 25, 26, 38, 39, 40).

5.5.1 Simplex Configuration.

The simplex configuration is a single dedicated computer that may have one or more sensors/subsystems connected to input/output ports as well as actuators connected to the output ports. The exchange of data is totally controlled by the computer using its internal computer bus.

5.5.1.1 Autonomous Fault Detection (In-Line Monitoring) for Simplex Configuration.

The principal fault detection technique used with these configurations consists primarily of the data validity checks previously discussed. In addition to those techniques, the use of built-in test equipment for self-monitoring is essential to the simplex system. In most instances, the reliability of the self-monitoring concept is less than that required for critical functions and as a result, simplex systems have not been widely accepted configurations for other than essential or nonessential functions.

5.5.2 Single-String Configurations.

The single-string configuration makes use of digital data buses to interconnect simplex configurations in which the computer performs dedicated functions that require exchange of data in order to implement all possible modes for the dedicated functions. Figure 5-11 can be considered a single-string system in that the flight management computer unit (FMCU) is a single unit interconnected by digital data buses to the sensors which collectively provide signals required to determine the aircraft's state (position, velocity, altitude, attitude) at any moment in time.

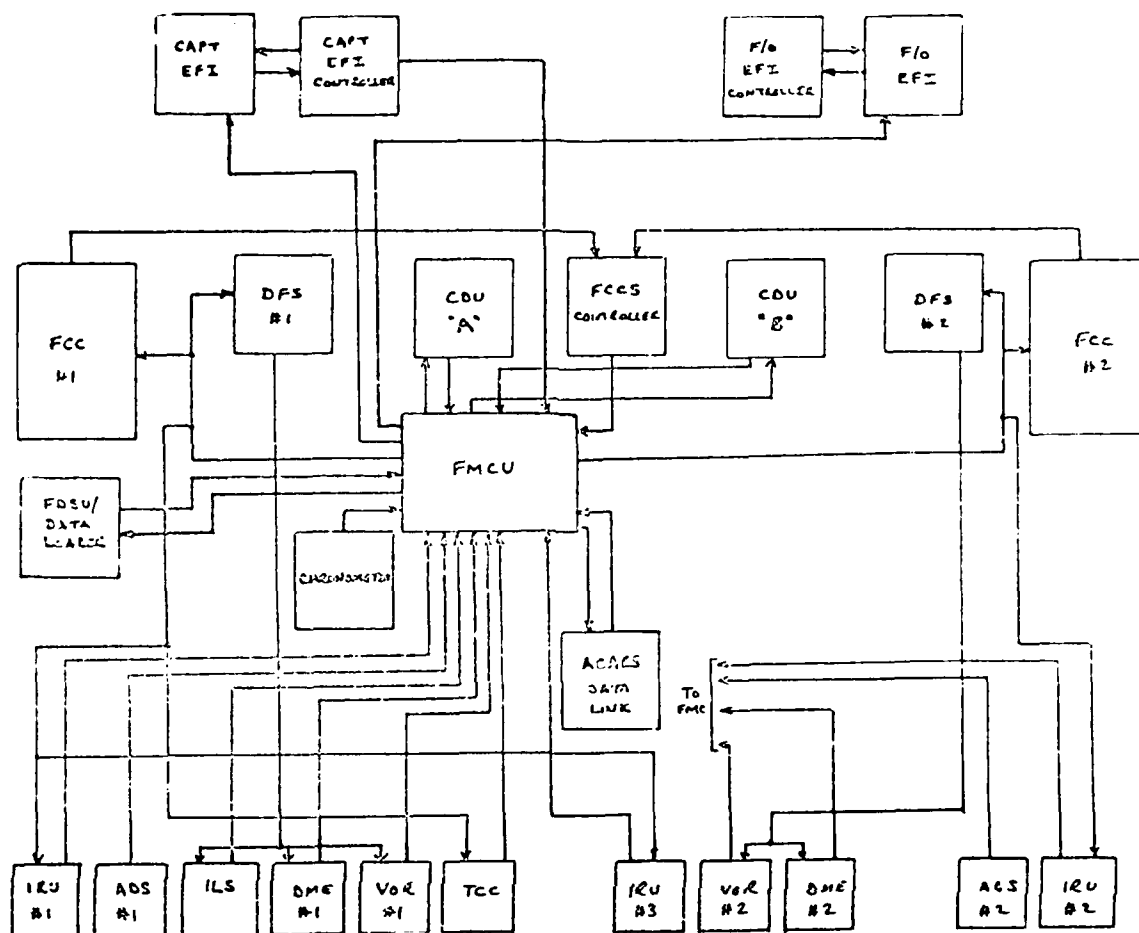


FIGURE 11. FLIGHT MANAGEMENT COMPUTER SYSTEM CONFIGURATION 2:
SINGLE SYSTEM/DUAL CDU INSTALLATION

5.5.2.1 Fault Detection and Error Management. The single-string systems utilize not only the data validity checks previously discussed, but also make use of the data transmission error detection and the processor failure detection concepts previously discussed. These systems may also make use of analytical redundancy in which the measurements provided by faulty sensors may be synthesized from an analytical model relating these measurements to measurements from "known good" sensors (references 7, 8, and 39).

5.5.3 Dual System Configuration.

Dual system configurations are typified by the configuration given in figure 5-12. these systems typically can experience a failure in a single channel and still provide the required system functions.

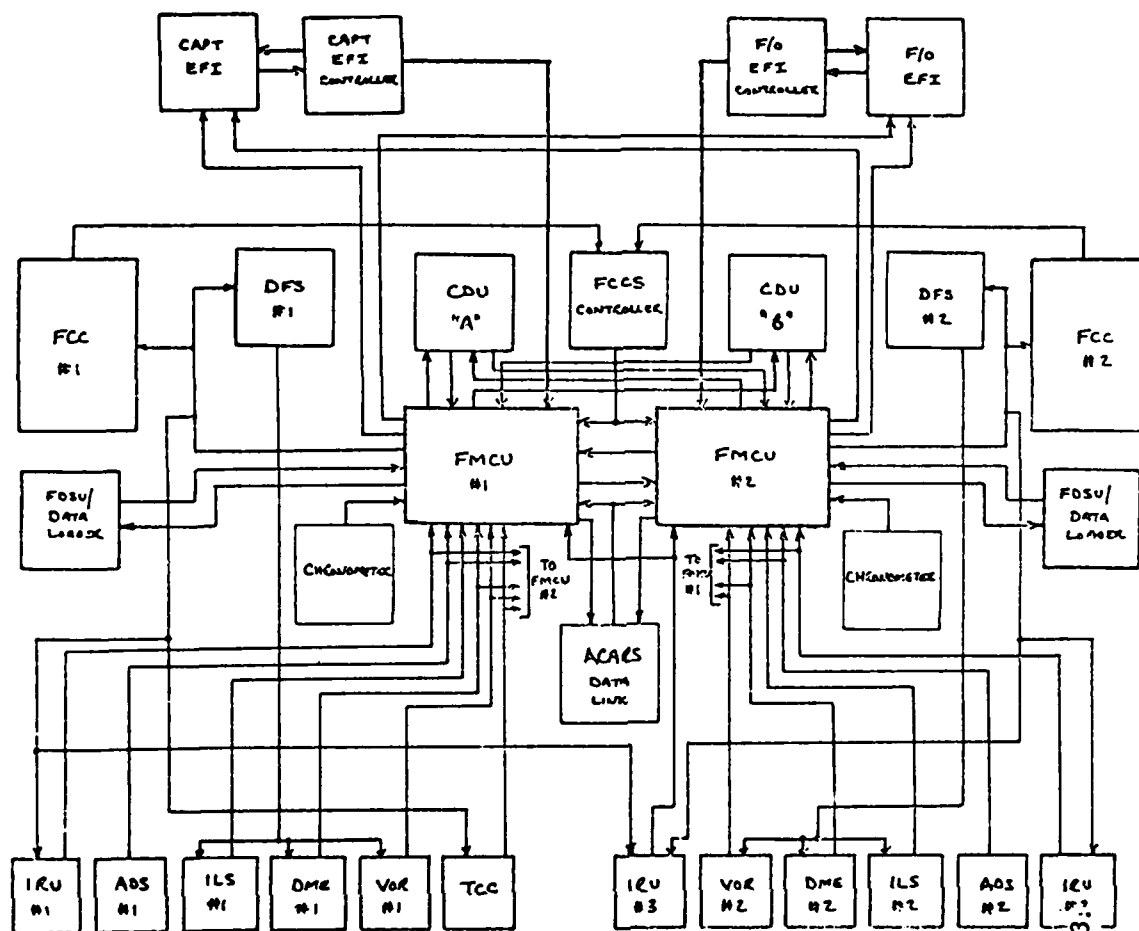


FIGURE 12. FLIGHT MANAGEMENT COMPUTER SYSTEM CONFIGURATION 3:
DUAL SYSTEM INSTALLATION

5.5.3.1 Fault Detection and System Error Management. Dual system configurations typically make use of the processor failure detection, data transmission error detection, and data validity tests previously discussed. In addition, the outputs of the two identical channels can be compared. Such a comparison can only determine that one of the systems is at fault if the outputs do not agree within some tolerance limit. Depending upon the degree of cross-strapping between the systems, various fault detection and isolation tests can be performed on the sensor data. Reference 39 discusses inter-unit switching of the dual digital redundancy-managed flight control system shown in figure 5-13. "In normal operation, the string of components in system A (or B) control the airplane with the off-line systems standing-by in reserve. Built-in test equipment (BITE), software tests, and end-around tests continuously monitor the critical components of each system for proper operation. Detection of a malfunction may be by component BITE, software comparison and data reasonableness tests on each component, or by sensing of overall aircraft response. If the performance of an element in system A falls below preselected levels, it is automatically replaced by system B counterpart. The system performance is thereby maintained in the presence of multiple failures as long as the failures do not affect like components to both systems. At the system level a redline monitor continually examines certain airplane parameters to detect

critical flight conditions. If critical values and vehicle attitude, attitude rate, airspeed, Mach number, vertical acceleration, or angle-of-attack are exceeded, the redline monitor automatically transfers from system A to system B. System B devices, which have previously been declared failed, are not placed on-line when the transfer takes place" (reference 39).

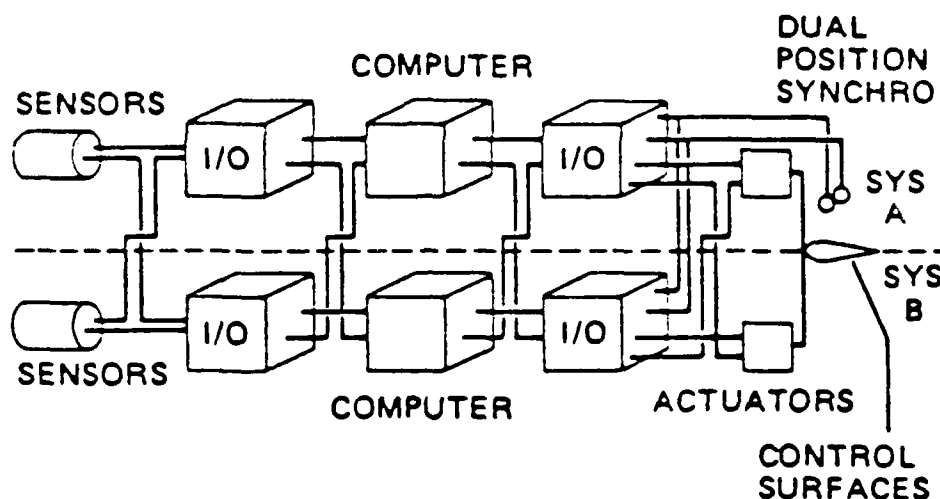


FIGURE 13. DUAL DIGITAL FLIGHT CONTROL SYSTEM
WITH INTER-UNIT SWITCHING (Ref. 39)

Reference 39 discusses the deficiencies of this redundancy management method based upon the results of extensive simulation. These deficiencies included:

(1) An airplane maneuver must be conducted to detect a failure of rate gyro and accelerometer sensors by monitoring the sensor output.

(2) Redundancy management allowed use of both the DME and ILS glide-slope models simultaneously. Since each model is dependent on valid data from the other set of sensors, an interlock should be incorporated to prevent both models being on-line at once.

(3) Sensor reasonableness limits and fault timers on the sensors used in the system are also affected by airplane dynamics. Certain types of faults; e.g., latent (failure to zero), bias, slow drift, and scale factor shift, are not readily detectable. Sensor noise has an adverse, but as yet undetermined, affect on the performance of the redundancy management algorithms which will force an increase in the various detection and isolation thresholds in order to avoid nuisance trips. Reference 39 discusses use of some form of bias compensation (equalization) proposed for triplex systems which could be applicable to the dual configuration.

Reference 39 also discusses some problems associated with failure detection and off-line/on-line role swapping of the computer units because the input/output computers were running asynchronously. Problems experienced with digital computer units, when a role swap was required, could be solved by either abandoning "the concept of entirely different code execution in the off-line machine and let the off-line digital computer unit process input/output unit data as if it were on-line or send enough data to the off-line digital computer unit to ensure that the swap would be nearly transparent" (reference 39).

An additional problem encountered was in the initialization of the on-line and off-line computers. Reference 39 discusses various tradeoffs that need to be performed for dual configurations in which one processor is active and the other is standby.

5.5.4 Dual-Dual System Configuration

Figure 5-14 depicts a typical dual-dual automatic flight control system configuration. While this may appear to be a quad system in that four computers are shown, the system is typically operated as two dual/dual fail-operative systems (reference 45). In the dual-dual configuration, each flight control computer has two channels which exchange data within the computer. Data exchange also occurs between flight control computer 1 and flight control computer 2 as shown in figure 5-14.

5.5.4.1 Fault Detection and System Error Management. In addition to the processor failure detection, data transmission error detection, and data validity tests previously discussed, the dual-dual system can make use of comparison monitoring in which the outputs of the channels can be either compared on a bit-by-bit basis or a differential basis. Bit-by-bit comparisons are predicated on the assumption that the output will be in agreement except in the case of a fault. Differential comparisons, on the other hand, do not require perfect agreement between the channels but instead permit a certain amount of skew (reference 18). "There are several techniques available for ensuring that all channels have identical inputs even though the sensors have nonidentical outputs due to skew and tolerance effects. The method that is best suited to this configuration, since it can be serially and in a single pass, is median selection" (reference 18).

Cross-strap controllers that make use of median selection algorithms to obtain the control signal require a minimum of three channels for the concept of median selection to work. The major flaw with median selectors is the transient that occurs, should the source of the median signals suddenly fail. The median selector immediately switches to another source allowing a transient to occur. The size of the transient depends upon how far the new median is from the old median.

"There is no clear cut directive as to what should be done about the defective signal when a failure occurs. A primary consideration is whether it should be switched out. If it is switched out, a strategy must be implemented to ensure that three or more inputs remain. If it is to remain as an input, a decision must be made as to whether the signal should assume any value it wants, or whether it should be forced to a particular value and what this particular value should be. If it is forced to an extreme value, a subsequent failure can result in the faulty signal being selected as the median. If it is forced to zero, the small signal behavior in the remaining controllers may be erroneous. In general, the decisions made in the above considerations will be dependent upon the application and the user's priorities" (reference 18).

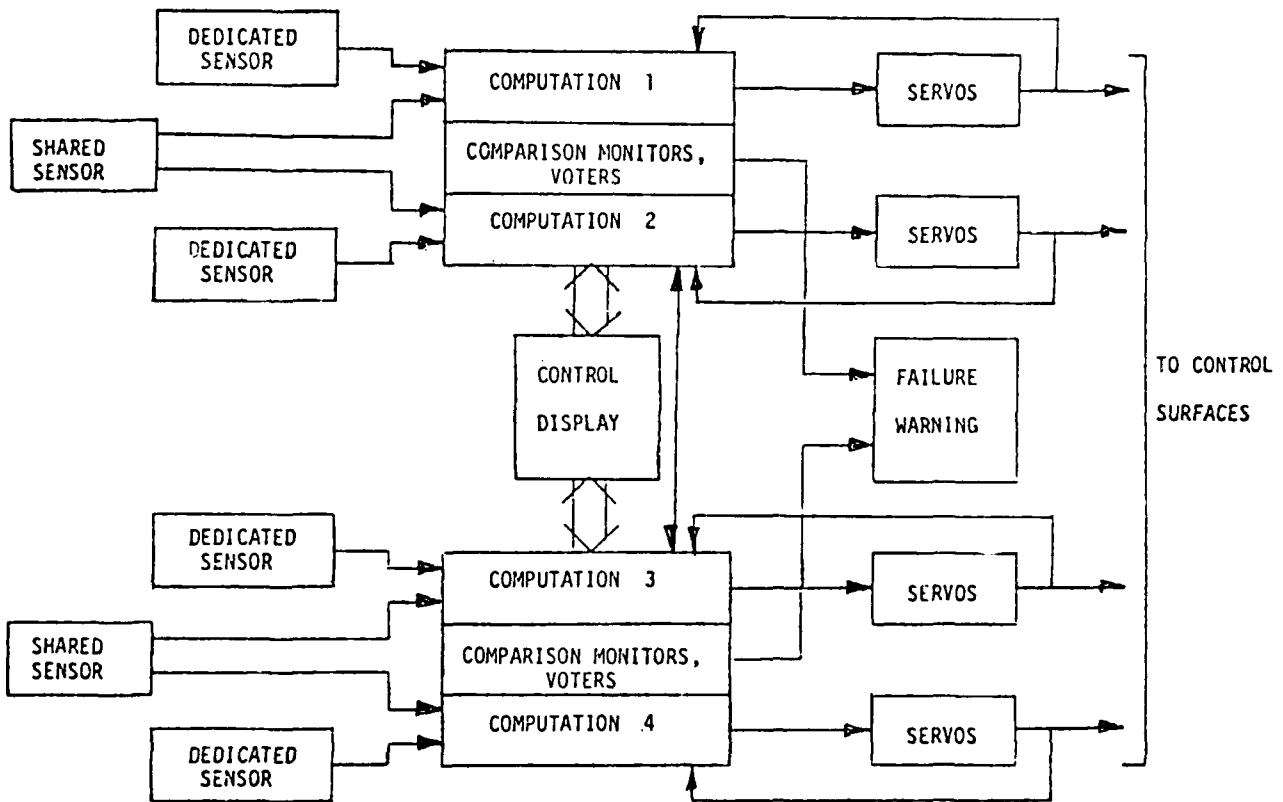


FIGURE 5-14. TYPICAL DUAL-DUAL AFS CONFIGURATION

"As indicated previously, median selection cross-strapping requires a minimum of three channels to survive a failure in one controller and provide undegraded performance. A minimum of four channels is required to survive two failures. There is no median as such for four signals. What is usually done is to take either the more negative or more positive of the two inner signals. An alternative would be to operate in an active-standby mode until the first failure occurs" (reference 18).

"Another method that is commonly used to develop the control signal in cross-strapped controllers is an averaging cross-feed. In this method, each channel accepts inputs from all other channels and computes their average which is then used as a signal" (reference 18). Only two channels are required to provide undegraded performance after a single failure if autonomous fault-detection techniques are utilized. A third channel would allow two failures. An additional channel, bringing the count to three and four, respectively, would be required if comparison monitoring were used for fault detection (reference 18).

"Certain classes of digital controllers develop the control signal in yet a third way. In this method, the channels are interconnected, and the resulting signal is derived from a majority vote of all inputs on a bit-by-bit basis. The digital signal's bit can only have two values: Either a one, or a zero. No other values exist. To use this method, certain criteria must be satisfied:

- (1) There must be at least three channels in order to get a two of three (four channels for a three of four).
- (2) In the absence of failures, all outputs must normally agree on a bit-by-bit basis.
- (3) The redundant channels must operate in time synchronism to some extent to facilitate bit-by-bit voting" (reference 39).

Additional cross-strapping in a dual-dual system is possible, including cross-strapping similar to that depicted for the dual system in figure 5-13. References 18 and 39 describe many of the tradeoffs involved with different cross-feed and data transmission alternatives. The flight safety requirements and the function criticality requirements of reference 44 must be taken into account in the evaluation of any of the different error management concepts.

5.5.5 Triplex.

Figure 5-15 depicts a typical triplex system configuration. In this configuration, data are exchanged between the three flight control computers prior to output to the control surfaces.

5.5.5.1 Fault Detection and System Error Management. The processor failure detection, data transmission error detection, and data validity tests are typically used in triplex configuration. Majority voting logic is typically used for the processing of sensor inputs and comparison of cross-strapped data exchanges. As depicted in figure 5-15, three actuators are provided for each control surface. The outputs of the redundant driver or surface actuators may be summed or combined in several ways: position, velocity, or force (reference 18).

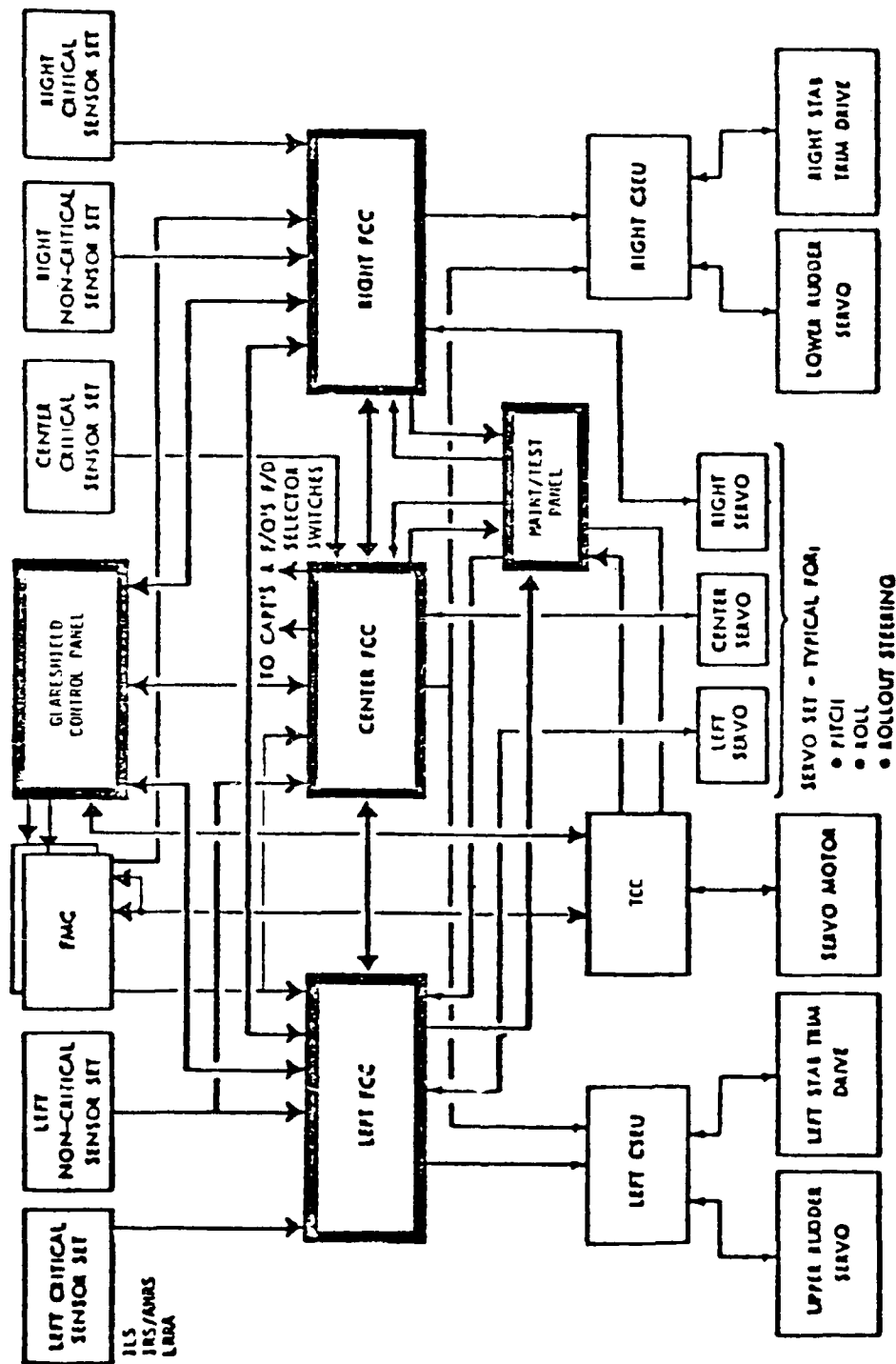


FIGURE 5-15. TYPICAL TRIPLEX DAFS ARCHITECTURE

Position-summing systems have three inherent weaknesses:

(1) Data transients from hard-over failures in any one channel are simply transmitted through to the output.

(2) A failed channel actuator must be disengaged to some predetermined position immediately.

(3) Positional gains and authority to the remaining channel actuators must be increased once a failure has been detected and corrected if the output positional relationship is to remain the same (reference 18).

A variation of position summing is the use of redundant, or "split" surfaces with a single channel integrated actuator. This requires that failure of a given channel or actuator results in the disengagement of the control surface to a streamline or neutral position (reference 18). The criticisms that apply to position summing typically apply also to velocity summing concepts.

Force summing has been the most popular method (references 18 and 45). Variations on the force summing concept according to reference 45 are active/standby and active/on-line. As stated in reference 45, "the choice of redundancy technique will depend to a large degree upon the specific aircraft configuration."

5.6 ARINC SUBSYSTEMS.

The various Aeronautical Radio Incorporated (ARINC) characteristics pertaining to digital systems and the sensor subsystems which interface with these systems contain material related to system design and performance requirements. While these documents primarily pertain to commercial transport type aircraft, they provide meaningful information as to the functions/modes, accuracy and format of output signals, and other related requirements for these systems and subsystems.

The ARINC characteristics for area navigation systems, flight control computer systems, flight management computer systems, and thrust control computer systems provide design guidance necessary to achieve a certain minimum level of operational capability and prescribe the basic functions these systems shall have as well as modes of operation of these systems and the method of automatic reversion when sensor data are invalid. This section of the book provides only a synopsis of the relevant ARINC characteristics. The user of the handbook should make sure that he has available the latest published version of each characteristic for those systems and subsystems which require validation or interface with a subsystem which requires validation.

5.6.1 Area Navigation System.

5.6.1.1 ARINC Characteristic 581, Mark 1 Air Transport Area Navigation System.

The ARINC 581 system is a station-oriented three-dimensional navigation system whose primary position-determining data sources are baro-corrected altitude, VOR bearing, and DME distance relative to a selected VORTAC or colocated VOR-DME station. The waypoints are defined in terms of bearing (theta) and distance (rho) from the referenced ground facility and an altitude or elevation above sea level (reference 46).

This system provides five basic functions listed in table 5-2.

TABLE 5-2. ARINC CHARACTERISTIC 581 FUNCTIONS

Function	Description
Position Determination	Bearing/distance computed to (or from) waypoint whose own position is defined in terms of its bearing and distance from a VORTAC or colocated VOR-DME.
Horizontal Guidance Signals	Cross-track deviation from a desired track which extends to or through the waypoint.
Parallel Track Navigation	Generate desired parallel track offset at a selected distance either left or right of the original track.
Vertical Guidance	Vertical track deviation from a selectable vertical ascent or descent angle to reach a desired altitude either at the waypoint or at some desired distance from the waypoint as determined by the vertical track offset.
Altitude Alert	Alert the pilot upon approaching the desired altitude by sequence of both visual and aural signals in sufficient time to establish level flight at that desired altitude.

Table 5-3 provides the range and resolution of the inputs for the minimum capability ARINC 581 system.

TABLE 5-3. ARINC 581 SYSTEM INPUTS — RANGE AND RESOLUTION

Input	Range	Resolution
Waypoint Distance (rho)	0-199.9 nautical miles	0.1 nautical miles
Waypoint Bearing (theta)	0-359.9 degrees	0.1 degree
Desired Altitude	0-50,000 feet	1 foot
Desired Track	0-359.9 degrees	0.1 degree
Cross-Track Offset Distance	0-20 miles	1 mile increments left or right
Vertical Track Angle	0± 9.9 degrees	0.1 degree
Along-Track Offset Distance	± 99 miles relative to the waypoint (+ = beyond; - = ahead of)	1 mile
Active Waypoint Number	Minimum of six	--
VOR Frequency	108.00-117.95 MHZ	50 KHZ

The operating modes selectable on the CDU include off, en route, terminal, approach, and test.

The 581 characteristics specifies the ARINC system operation in the event of input sensor failure. These are summarized in table 5-4.

The RNAV equipment accuracy is defined as that given in FAA Advisory Circular 90-45A. The input sensors signal accuracies are given on later pages of this section.

TABLE 5-4. ARINC CHARACTERISTIC 581 SYSTEM OPERATION WITH FAILED SENSORS

Input Failure	System Operation Mode
VOR Sensor	Complete loss of RNAV
DME Sensor	Complete loss of RNAV
Altitude Input	Loss of slant range correction, vertical navigation, and altitude alert
True Airspeed	Design Dependent
Supplemental Navigation System	Design Dependent

5.6.1.2 ARINC Characteristic 583-1, Mark 13 Area Navigation System. The ARINC 583 system is an earth-oriented three-dimensional navigation system which processes sensor inputs using a spherical or other accepted earth model, in contrast to the ARINC 581 system whose computations are referenced to the VORTAC or VOR/DME "station" being received. The 583 system employs bearing and distance data derived from ground-based VORTACS and colocated VOR/DME facilities for lateral navigation data computation. "Altitude, derived from an onboard air data computer or similar source, is employed for a slant range correction of DME distance and for vertical navigation computation. Additionally, the system is capable of accepting data from an inertial sensor in an 'INS/ISS-supported' configuration and of complementing an ARINC 561 INS in an 'INS-dependent' configuration.

"In installations where inertially derived information is available, the system will determine aircraft track by a VOR/DME/INS mix computation, with automatic reversion to INS when the VOR/DME data are invalid. When INS information is not available, or is invalid, VOR/DME fixing will be employed with air data/magnetic heading aided smoothing. In this case, automatic reversion to air data-based DR navigation will occur when the VOR/DME information is invalid.

"Navigation and steering command signals will be computed with respect to individual great circle 'legs,' each defined by two waypoints. Each way-point is defined by its altitude or elevation above sea level and either its latitude and longitude or its bearing (theta) and distance (rho) from a reference VOR/DME facility on the ground" (reference 47).

The 583 system provides the basic functions listed in table 5-5. Table 5-6 provides the range and resolution of the inputs for the minimum capability ARINC 583 system. The 583 characteristic specifies the ARINC system operation modes in the event of input sensor failure. Some of these modes are summarized in table 5-7. A dual VORME installation provides many possible modes in the event of failure of a single sensor as shown in table 5-7.

TABLE 5-5. ARINC CHARACTERISTIC 583 BASIC FUNCTIONS

Function	Description
Position Determination	Latitude and longitude. Bearing/distance computed to next waypoint.
Horizontal Guidance Signals	Cross-track deviation and cross-track distance from a desired track defined by the line joining two successive waypoints.
Parallel Track Navigation	Generate a desired parallel track offset at a selected distance either left or right of the original track.
Vertical Navigation	Flight path angle should be automatically computed as a function of the distance between consecutive waypoints and the difference between their altitudes. The computed angle should emanate from the wayline of the "to" waypoint.
Vertical Guidance	Vertical track deviation from a computed or pilot selected vertical ascent or descent angle to reach a desired altitude either at the waypoint or at some desired distance from the waypoint as determined by the "along track offset".
Altitude Alert	Alert the pilot upon approaching the desired altitude by a sequence of both visual and aural signals, in time to establish level flight at that desired altitude.

TABLE 5-6. ARINC 583 SYSTEM INPUTS RANGE AND RESOLUTION

Input	Range	Resolution (minimum)
Waypoint-to-VORTAC Distance (rho)	0-399.9 nm	0.1 nm
Waypoint Bearing From VORTAC (theta)	0-359.9	0.1 degree
Waypoint Latitude	0-89° 59.9' N or S	0.1 minute of arc
Waypoint Longitude	0-179° 59.9' E or W	0.1 minute of arc
Desired Altitude	0-50,000 feet	1 foot
VOR/DME Station Elevation	0-15,000 feet	100 feet
VOR/DME Station Magnetic Variation	0-179.9 degrees E or W	1 deg.-manual input 0.1 deg.-program entry
VOR/DME Station Latitude	0-89° 59.9' N or S	1 minute of arc
VOR/DME Station Longitude	0-179° 59.9' E or W	1 minute of arc
Cross Track Offset Distance	0-20	1 mile increments left or right
Along Track Offset Distance	+ 99 miles relative to the waypoint (+ = beyond, - = prior to)	1 mile
Desired Vertical Flight Path Angle	0 ± 9.9 degrees	0.1 degree
Active Waypoint Number	Minimum of 20	

The RNAV equipment accuracy is defined as that given in FAA Advisory Circular 90-45A. The input sensors signal accuracies are given on later pages of this section.

TABLE 5-7. EXAMPLE OF ARINC CHARACTERISTIC 583 SYSTEM OPERATION WITH FAILED SENSORS

	System Operating Mode*								
Sensors Failed	1	2	3	4	5	6	7	8	9
INS	x					x	x		
VOR 1		x				x	x		x
DME 1							x		x
VOR 2				x		x			x
DME 2					x				x
BARO ALT								x	
True Airspeed									
Mag Heading									

*Mode 1,7 - VOR/DME RNAVNAV
 Mode 2,4 - INS/VOR/DME RNAV/VNAV
 Mode 3,5 - INS/VOR/DME RNAV/VNAV
 Mode 6 - Dead Reckoning with Air Data
 Mode 8 - RNAV
 Mode 9 - INS RNAV/VNAV

5.6.2 Flight Control Computer System.

5.6.2.1 ARINC Characteristic 701, Flight Control Computer System. The Flight Control Computer System (FCCS) is a portion of the larger Automatic Flight System (AFS) specifically designed to provide autoland capability for commercial transport aircraft. Figure 5-16 (attachment 1, reproduced from reference 48, ARINC Characteristic 701) depicts the basic AFS architecture. The Flight Augmentation Computer System (FACS) is not shown in figure 5-16 since the functions it provides to assure suitable aircraft handling qualities for manual control are "obviously unique to a particular airframe type and may include (but are not necessarily limited to) pitch trim, stability augmentation, aileron gain programming and flap load limiting" (reference 48). The FCCS include the Flight Control Computer Unit(s) (FCCU), the controller, and annunciator unit, the Control Wheel Steering (CWS) force sensor, and the wheel disconnect buttons.

"The functions of the FCCS are:

a. Attitude Hold - the system using gyro data provides attitude hold in pitch, roll, and/or heading when pilot is not exerting a force on the control wheel and no superseding mode has been selected. The mutually exclusive roll attitude or heading hold is determined by the roll attitude existing at the time the pilot ceases to exert roll force on the control wheel.

AFS ARCHITECTURE

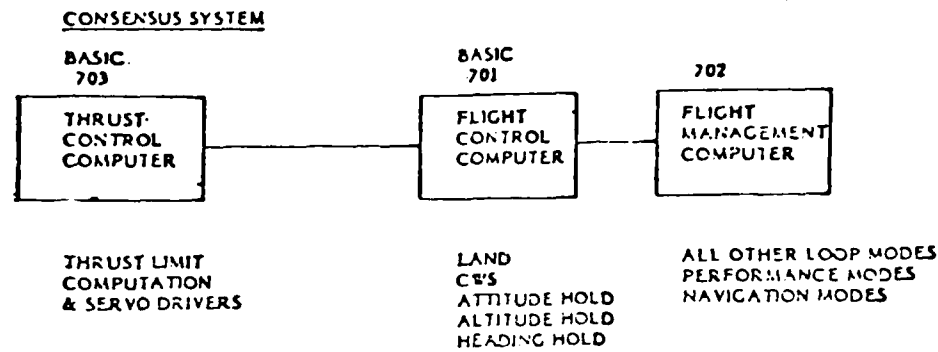


Figure 1 - Basic System (no channelization shown)

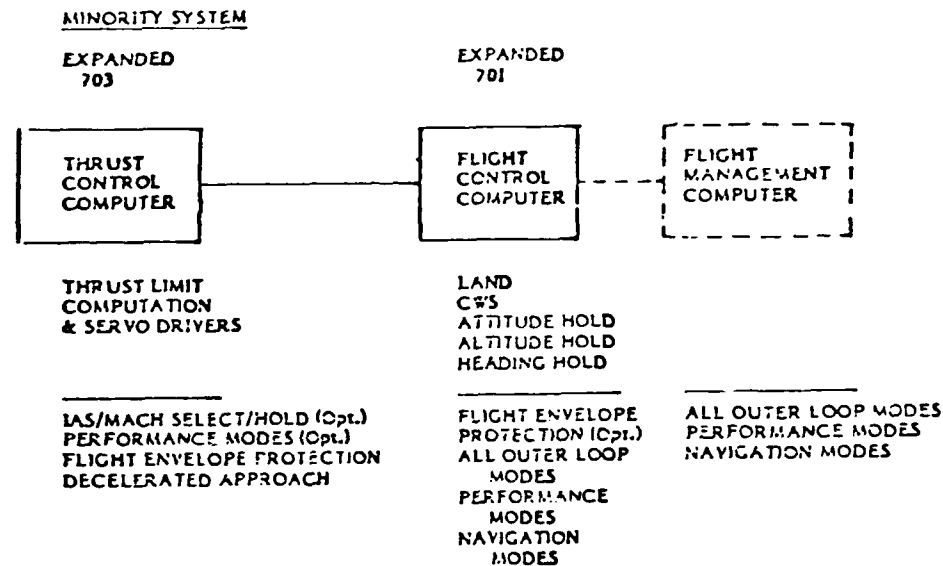


Figure 2 - Expanded System (no channelization shown)

	Basic Configuration FCCS	Basic System (Figure 1)	Expanded System (Figure 2)
1	Dual Channel - Intended as a subset of a dual/dual fail-operative system (2 FCCU installed) or may be: a) Single Unit (FCCU) fail passive system. b) Single Unit with inline or limited monitoring. c) Simplex (unmonitored).	Require independent FMC. Restricted mode list in FCC. Loss of FMC results in degraded capability. Interface is a selectable subset of the expanded version.	Unlimited mode list in FCC. All functions retained if no FMC. May be operated in conjunction with a FMC if desired, although mode duplication will occur.
2	Triplex - Intended as a subset of a Triplex fail-operative system (3 FCCU installed).	same as above	same as above

Notes A. It is recognized that a "pure" dual/dual FCCU may not exist as various monitoring schemes may be implemented which do not require a split channel FCCU. However, the AFS Subcommittee felt that the internal monitoring differences may all be incorporated within the dual/dual interface definition.

B. The AFS Subcommittee felt that the triplex configuration is viable only if triple channels are installed. A two channel fail-passive version of the Triplex configuration would more properly be handled by configuration 1.

FIGURE 5-16. ARINC 701 INTERFACES

b. Heading Hold - the system provides a heading hold function when roll force on the control wheel is below breakout and existing roll angle is below a predetermined level (nominally between ± 7 degrees).

c. Control Wheel Steering - the system provides a means of introducing manual commands to the FCCS through the application of forces to the control wheel in pitch and roll. Force transducers detect and transmit commands by the pilot which supersede any other mode selected, and instructions to the system with the exception of LAND mode. These signals are computed together with inertial parameters in the FCCU to provide the intended pitch and roll maneuvers to attain the desired attitude hold and/or heading hold condition of the aircraft.

d. Altitude Hold - this function provides a 'clamping' of the preexisting barometric altitude as sensed by the Air Data System and computes the necessary commands to the elevator servo using altitude error signals developed in the FCCU, baroaltitude rate from ADS, pitch and roll angles' plus vertical acceleration from the IRS or AHRS.

e. Automatic Approach and Landing - the FCCS provides, in conjunction with deviation signals from the ILS in addition to the ADS and IRS/AHRS signals, a navigation guidance function to place the aircraft on the runway. A radio altimeter input is additionally required except for ICAO Category I capability.

"When LAND mode has been selected by the pilot on the AFS Controller after having selected the proper ILS channels and having dialed in the proper runway heading, the FCCS will cause the aircraft to bracket the Localizer and glide slope beams of the ILS using error signals from ILS receiver and runway set heading error from the AFS Controller.

"The radio altimeter signals are used by the FCCS for gain programming and to initiate 'flare' and 'runway align' maneuvers during the approach and land sequence. The Localizer error signal is used during the 'Roll out' on landing, the Glide slope error signals from the ILS are not used during or after the flare maneuver.

f. Autothrottle - the autothrottle function of the basic 'autopilot' provides thrust control during the automatic approach and landing phase of the aircraft. When LAND mode has been selected, and the AUTOTHROTTLE MODE switched 'on,' the FCCS will cause the aircraft to fly at the airspeed selected on the AFS Controller until overridden by the pilot which causes the AUTOTHROTTLE mode switch to automatically drop to 'off.'

g. Missed Approach - when GO-AROUND mode has been selected (during the approach and landing phase) a programmed pitch angle and wings level attitude is commanded with simultaneous full advancement of the auto throttle if that mode has been selected.

h. Back Course Localizer - when selected on the AFS controller, the FCCS will control the aircraft in the turn axis only (pitch modes not affected) providing means to fly the localizer beam on approach to land the aircraft on a non-glideslope instrumented back course runway.

i. Flight Director Signals - the FCCS provides pitch and roll steering signals for display on the Flight Director allowing the pilot to fly the modes of

the AFS when the flight servos are not engaged. This display additionally allows the pilot to monitor the operation of the system when the AFS is engaged. Additionally, when AUTOTHROTTLE MODE is engaged, a throttle demand error signal is provided for flight director display.

j. Flight Envelope Protection - regardless of the engaged mode, the AFS (FCC) should at all times control the aircraft to ensure that the VMO will not be exceeded, and with the exception of certain windshear situations (see Commentary below) the aircraft will not be operated below minimum speed (i.e., alpha speed, minimum maneuvering speed, etc.) if there is a single source of minimum speed."

COMMENTARY

"Approach and go-around automatic and flight director systems for transport aircraft have been designed in the past to provide smooth precise control and performance for winds that are represented by statistical models. The winds resulting from the models are relatively moderate in terms of shear and turbulence, and are typified by wind models in FAR Advisory Circulars. In the past few years, a related but seemingly independent phenomenon has been identified as contributing to approach and go-around incidents and accidents — low-level windshear. Such wind environments have ranged up to severe, where all of the aircraft's potential performance capabilities must be called upon and utilized to avoid an incident. Designers of new autopilot, flight director, and autothrottle equipment and associated control laws should be aware of this wind phenomenon and not limit the system designs for handling such wind conditions.

The system modes that are affected include localizer track, glideslope track, runway align, flare, and speed-hold during approach and landing. For the takeoff and go-around problem, it has been shown that it may be necessary to fly the aircraft at high pitch attitudes and below the normal reference speeds during some wind encounters. Such logic has never been built into a takeoff or go-around autopilot or flight director, but may be desirable to allow the flight crew to squeeze the last ounce of performance out of the aircraft. Special selectable modes should not be provided but normal modes should be 'windproofed.' Therefore, no pilot selection should be required during a windshear encounter. Hence, during more typical operations, the crew should notice no difference in system functional characteristics when compared to previous systems" (reference 48).

"This section provides a list of functions which would be assigned to the FMCS if the basic FCCS is adopted, but which would normally be incorporated into the expanded FCCS if that system were adopted. Functions are:

a. Vertical Speed - using data from the ADS and/or IRS, vertical speed may be commanded by selection on the AFS controller. Pitch error commands are computed for the FCCS.

b. Heading Select - a preset heading may be set by the controller and initiated by a momentary switch on the AFS controller. After the preset heading is attained, the MODE annunciator indicates termination of the Heading Select mode. Roll rate/acceleration limiting may be needed to control these parameters in this preset function of the FMCS.

c. Mach Hold - this function may be initiated by a switch on the AFS controller and is provided by computing the error signal in the ADS/IRS for control of mach by error commands to the FCCS/TCCS.

d. Airspeed Hold - this function is provided by providing a synchronized error signal of airspeed to the FCCS/TCCS.

e. Airspeed Select - this function is provided as preset command of airspeed which provides an error signal to the FCCS/TCCS. As in any preset command, limiting of dynamic parameters (pitch angle, normal acceleration) is necessary for passenger comfort and structural integrity of the aircraft.

f. Takeoff - this operational function provides various takeoff aids such as takeoff data computations, speed command profiles, and/or thrust commands. It may be a display function only or be combined with flight director and/or possibly autopilot control.

g. Altitude Select - this function is provided by the FMCS through selection of desired altitude on the AFS controller. Upon arrival at the selected altitude, the system automatically levels off and transitions to altitude hold. Visual or audio warnings alert the crew when the aircraft is approaching or deviating from a preselected altitude. The altitude alert function operates independently of 'autopilot' functions."

COMMENTARY

"Only the FMCS autopilot functions have been described in this section. The FCCS plus the FMCS described here provide the traditional basic autopilot functions. This need has arisen because airlines have had difficulty in standardizing their requirements for FMCS type functions. These functions will be included with more esoteric functions of radio navigation, energy management, and the like in the FMCS described by ARINC Characteristic 703" (reference 48).

The controller, often mounted on the glareshield, should typically provide the crew with the capability to control the FCCS, the FMCS, and the TCCS as given in table 5-8.

5.6 2.1.1 Controller Digital Data Inputs and Outputs. The following paragraphs list the digital data input and output ports needed on the controller and describe the information elements crossing the interfaces. Refer to ARINC Specification 429, 'Mark 33 Digital Information Transfer System (DITS)' for complete information on the digital data transfer method to be employed.

TABLE 5-8. ARNIC 701 FLIGHT CONTROL COMPUTER (FCC) SYSTEM
CONTROLLER FUNCTIONS

FUNCTION
Selection of Flight Director Display
Engage/Disengage Each or Disengage All Flight Control Computers from their Control Actuators
Arming of Autothrottle Function
Selection of Turbulence Mode
Command Hold of Existing Airspeed
Selection/Engagement of Desired Indicated Airspeed
Command Hold of Existing Mach Number
Selection/Engagement of Desired Mach Number
Selection/Engagement of Desired Heading
Selection of Bank Angle Limit
Command Hold of Existing Altitude
Selection/Engagement of Desired Altitude
Selection/Engagement of Desired Vertical Speed
Command Hold of Existing Vertical Speed
Engage FMCS Horizontal Navigation Mode
Engage FMCS and TCCS Vertical Navigation Mode
Command FCCS to Perform Coupled ILS, Localizer, or Back Course Approach
Selection of Course
Selection of Runway Heading*
Selection of NI/EPR Throttle Mode
Engage/Disengage Control Wheel Steering Mode**

*For any given setting of the runway heading selection facility, the outputs from the two sources of digital runway heading data should be identical. A check for this may be performed in the utilization equipment and the data rejected and an alert raised if a discrepancy is discovered.

**Assigned bits of the Discrete Word #2 allow the FCCS to revert to the basic mode in one axis (e.g. roll) while leaving the other axis (e.g. pitch) in the upper mode engaged. If CWS is in the basic mode in a given AFS mechanization, this feature allows for selectable split-axis CWS operation. (Reference 48)

5.6.2.1.1.1 Controller Digital Data Input Ports. Five digital input ports should be provided to accept data from two instrument buses, from two FCCS, and a third FCC or TCC.

"The ARINC 429 data words given in tables 5-9, 5-10, and 5-11 should enter and leave the controller via the ports described above. It is anticipated (but not required) that control information will be BNR encoded and that information for display will be BCD encoded. The low bit rate (12 - 14.5 kilobits per second) defined in Specification 429 should be used. Upon initiation, control words should be transmitted with not less than 3 iterations. However, receiver design should not be predicated on continuous reception of such control words. Latest mode selected prevails."

TABLE 5-9. FCCS DATA WORDS

WORD	LABEL	
	BNR	BCD
Selected Course #1	100	024
Selected Course #2	110	027
Selected Altitude	102	025
Selected Airspeed	103	026
Selected Heading	101	023
Selected Mach	106	022
Selected Vertical Speed	104	020
Selected Runway Heading	105	017
Selected N1/EPR	112	021
Discrete Word #1	270	-
Discrete Word #2	271	-

NOTE: As discussed in Section 5.7.19 (Reference 36), Selected Runway Heading words will be derived from two independent sources within the controller. The same label will be used in the words from both sources."

TABLE 5-10. FCCS DISCRETE INPUT AND OUTPUT WORD FORMATS
(DISCRETE WORD #1)

Bit No	Function	Encoding	
		Zero	One
1-8	Label		
9	Capt. Flight Director	Off	On
10	F.O. Flight Director	Off	On
11	Turbulence Mode	Not Requested	Requested
12	Autopilot #1	Not Engaged	Engaged
13	Autopilot #2	Not Engaged	Engaged
14	RESERVED (A/P #3)	Not Engaged	Engaged
15	Autothrottle #1	Not Armed	Armed
16	RESERVED (A/T #2)	Not Armed	Armed
17	Airspeed Hold Mode	Not Requested	Requested
18	Airspeed Select Mode	Not Requested	Requested
19	Mach Select Mode	Not Requested	Requested
20	Mach Hold Mode	Not Requested	Requested
21	Bank Angle Limit Select	See Below	
22			
23			
24			
25	Heading Select Mode	Not Requested	Requested
26	N1/EPR Select Mode	Not Requested	Requested
27	IAS on Throttle	Not Requested	Requested
28	Mach on Throttle	Not Requested	Requested
29	Spare		
30	Sign/Status Matrix Parity (odd)		
31			
32			

"The Flight Control Computer should accommodate three inputs from each of the shared sensor systems essential to the all-weather autoland operations and two inputs from the Flight Management Computer and other non-autoland-essential shared sensors."

TABLE 5-11. FCCS DISCRETE WORD # 2

Bit No.	Function	Encoding	
		Zero	One
1-8	Label		
9	Altitude Hold Mode	Not Requested	Requested
10	Altitude Select Mode	Not Requested	Requested
11	Vertical Speed Select Mode	Not Requested	Requested
12	Vertical Speed Hold Mode	Not Requested	Requested
13	Horizontal Navigation	Not Requested	Requested
14	Vertical Navigation	Not Requested	Requested
15	Land Command	Not Requested	Requested
16	LOC Approach Command	Not Requested	Requested
17	Back Course Approach Command	Not Requested	Requested
18	CWS #1	Not Requested	Requested
19	CWS #2	Not Requested	Requested
20	CWS #3	Not Requested	Requested
21	Pitch Upper Mode Cancel	Not Requested	Requested
22	Roll Upper Mode Cancel	Not Requested	Requested
23	Heading Hold	Not Requested	Requested
24			
25			
26	Spare		
27			
28			
29			
30	Sign/Status Matrix		
31			
32	Parity (odd)		

Notes: The SDI (Source Destination Identifier) is not utilized inasmuch as this function is provided by (dedicated) ports described in Section 5.8.2.

5.6.2.1.2 Bank Angle Limit Encoding. Bit Nos. 21, 22, and 23 of Discrete Word # 1 should be encoded to indicate selected bank angle limit as follows:

Limit	Bit No.		
	21	22	23
Not used	0	0	0
5°	0	0	1
10°	0	1	0
15°	0	1	1
20°	1	0	0
25°	1	0	1
30°	1	1	0
Spare	1	1	1

COMMENTARY

"A number of alternatives exists concerning the number and type of inputs to be provided in the Flight Control Computer for the various shared sensors. Unfortunately, the number of inputs that are actually needed depends to a great extent upon the system configuration and redundancy control selected by the airframe design team for the all-weather autoland capability. The debate between the dual in-line monitored system versus triple system concepts is not likely to be resolved in time for the 'next' airplane, if ever. Thus, it will be necessary to provide three inputs for each type of shared sensor system essential to the all-weather autoland functions.

"Operation of the Flight Management Computer will not be essential to the autoland functions. Further, the AFS Subcommittee feels that no more than two Flight Management Computer need provide only two inputs for data coming from the Flight Management Computers. The same logic indicates the need for only two inputs for those other non-autoland essential shared sensors" (reference 48).

5.6.3 Flight Management Computer System.

5.6.3.1 ARINC Characteristic 702, Flight Management Computer System. The Flight Management Computer System (FMCS) is a portion of the larger Automatic Flight System (AFS), previously depicted in figure 5-16. The FMCS interfaces with the Flight Control Computer System as shown in figure 5-17. The ARINC 702 FMCS comprises two units, the Flight Management Computer Unit (FMCU), and the Control/Display Unit (CDU). Three different configurations, a single system as shown in figure 5-17, a single system with two CDU, and a dual system are described in reference 49.

The basic system functions are given in table 5-12. The FMCU has 28 digital data input ports and nine separate, buffered digital data output ports. The FMCU has provision for 29 discrete input and output pins.

5.6.4 Thrust Control Computer System.

ARINC Characteristic 703 (reference 50) describes the functions of a Thrust Control Computer System (TCCS) and its interfaces with other components of the automatic flight system. The TCCS is made up of the components whose functions are listed in table 5-13. Figure 5-18 provides data on the TCCU interfaces.

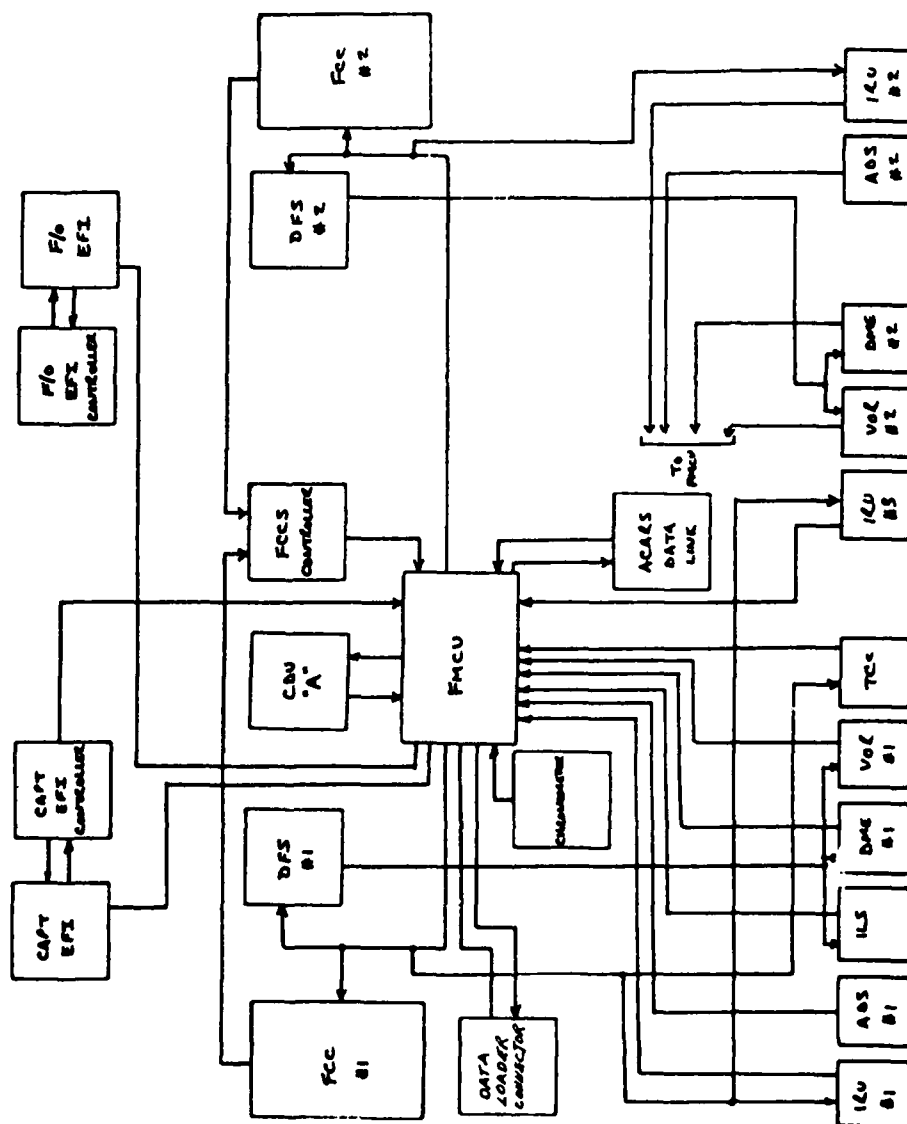


FIGURE 5-17. FLIGHT MANAGEMENT COMPUTER SYSTEM INTERFACE WITH FLIGHT CONTROL SYSTEM AND OTHER SUBSYSTEMS

TABLE 5-12. ARINC 702 FLIGHT MANAGEMENT SYSTEM FUNCTIONS

FUNCTION	SUBFUNCTION
Performance Management	Compute <u>Ground Distance and Time to Climb/Descend to Selected Altitude</u>
	Compute <u>Optimum Climb Profile Based on Selected Climb Mode</u>
	Compute <u>Optimum Cruise Altitude Based on Selected Cruise Mode</u>
	Compute <u>Maximum Range Based on Remaining Fuel</u>
	Compute <u>Maximum Endurance Based on Remaining Fuel and Selected Holding Speed and Altitude</u>
	Compute <u>Fuel Remaining Over Destination</u>
	Compute <u>Distance and Time-to-Go to a Computed Top of Descent Point</u>
	Compute <u>Engine-Out Performance</u>
Flight Planning	Waypoint Data Load
	Waypoint Sequence Load
	Crosstrack Offset for Parallel to Original Track
	Altitude Along Track Offset Distance
Lateral Navigation	Compute <u>Latitude/Longitude</u>
	Compute <u>Bearing/Distance to a Specified Waypoint or NAVAID</u>
Lateral Guidance	Compute <u>Cross-Track Deviation</u>
	Compute <u>Track Angle Error</u>
	Compute <u>Roll Angle Steering Command</u>
Vertical Navigation and Guidance	Compute <u>Vertical Flight Profile Altitude Deviation</u>
	Compute <u>Vertical Speed/Flight Path Angle Required to Maintain Vertical Flight Profile</u>
	Compute <u>Vertical Track Deviation</u>
	Compute <u>Vertical Steering Command (Vertical Speed, Vertical Acceleration, or Pitch)</u>
Speed/Thrust-Axis Control	Compute <u>Thrust Axis Limit Data and Mode Selection</u>
	Compute <u>N₁/EPR Commands for the TCC</u>
	Compute <u>CAS/Mach Commands for the FCC and the TCC</u>

TABLE 5-12. ARINC 702 FLIGHT MANAGEMENT SYSTEM FUNCTIONS (Continued)

<u>FUNCTION</u>	<u>SUBFUNCTION</u>
Electronic Flight Instrument (EFI) System Management	<u>Assemble and Format Data Required to Support EFI System Operation</u>
Data Update	<u>Load Data into FMCU from Data Loader</u>
IRS Initialization and Heading Set	<u>Output Pilot Entered (via CDU) Latitude/Longitude and Magnetic Heading to Initialize IRS</u>
Automatic Station Selection and Tuning	<u>Automatically Select VOR, DME, and ILS Channels in Accordance with Stored Program</u>
	<u>Tune Respective Receivers</u>
	<u>Enable Frequency Scanning and Frequency Diversity Modes of ARINC 709 DME</u>
Direct To	<u>Command a Direct Leg (Guidance) to Any Waypoint</u>
4-Dimensional Navigation	<u>Compute Flight Profile Time Deviation</u>
	<u>Computer Steering/Throttle Commands</u>
System Integrity Monitoring and Failure Warning	
Sensor Failure Warning Annunciation	

TABLE 5-13. THRUST CONTROL COMPUTER SYSTEM COMPONENTS AND FUNCTIONS

<u>COMPONENT</u>	<u>FUNCTION</u>	<u>MODE</u>
Thrust Control Computer Unit (TCCU)	N1 or EPR Limit Computation and Display Autothrottle	Speed Select and Hold Mach Hold and Adjust Thrust Rating a) Takeoff b) Climb c) Cruise d) Go Around e) Maximum Continuous Thrust f) Economy
		Flight Management Computer Control Mode Decelerating Approach
	Flight Envelope Protection	Throttle Pusher Minimum Speed Maximum Speed
	Engine Trimming Mode Selection	
Thrust Rating Panel		
Autothrottle Disconnect	Deselect Current Auto-throttle Mode	
Go-Around Initiation Switches	Initiate Go-Around	
Throttle Actuator Unit		
Throttle Clutch		

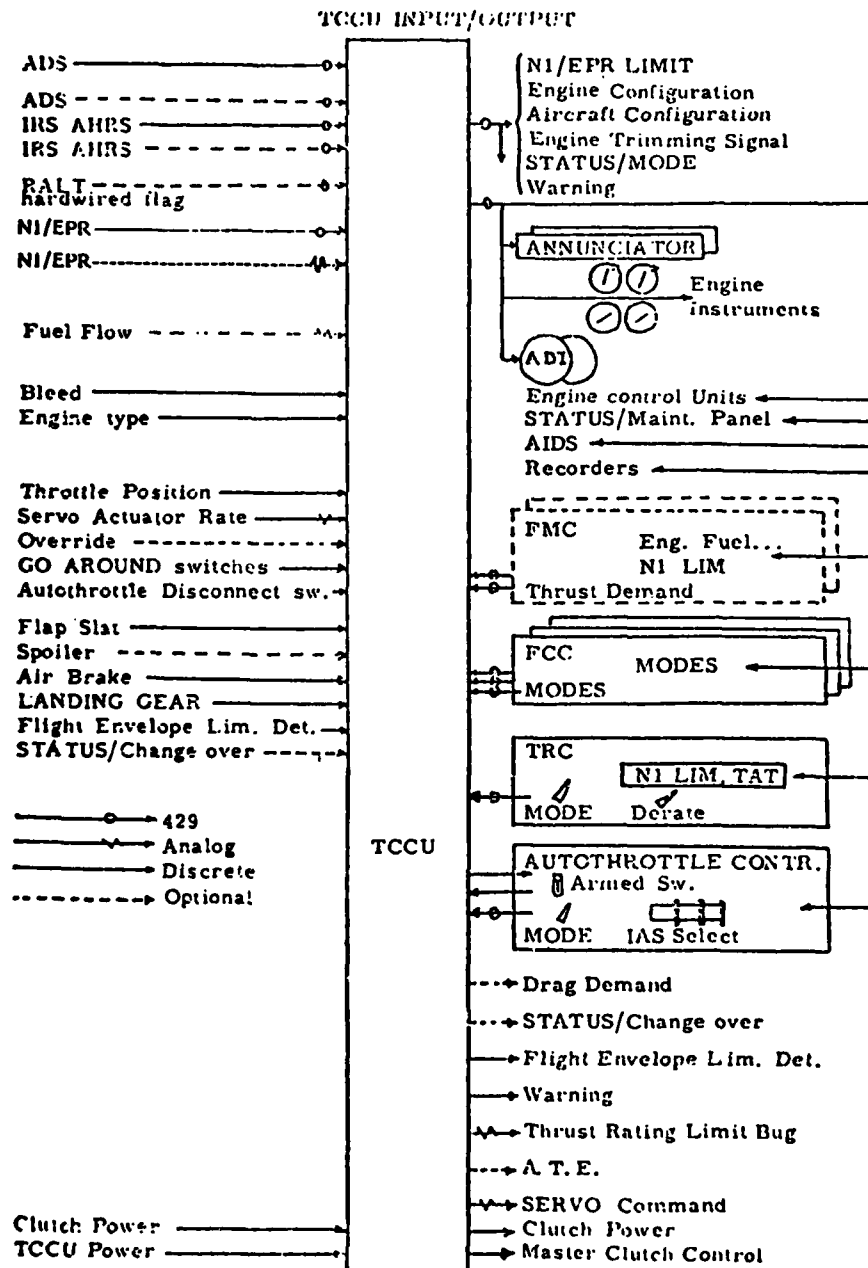


FIGURE 5-18 ARINC 703 THRUST CONTROL COMPUTER INTERFACE

5.6.5 Inertial Reference System (IRS).

ARINC Characteristic 704-3 (reference 51) describes the functions and modes of a local-level inertial reference system and its interfaces. Table 5-14 summarizes the IRS outputs and inputs. It should be noted that the IRS does not compute horizontal and vertical steering signals. Table 5-15 presents the digital control word/format. Figure 5-19 is a block diagram depicting the interfaces of the IRS with other avionics. Reference 51 recommends "that all data be output identical on all buses."

5.6.6 Attitude and Heading Reference System.

ARINC Characteristic 705-3 (reference 52) describes the vertical (normal and basic) and horizontal (magnetic and directional gyro (DG modes of operation of a local-level attitude heading reference system (AHRS). The AHRS requires input of digital true air speed (Label 210) 8 times per second from two air data systems (ARINC 706) as a minimum. Altitude (Label 203) is used to compute flightpath angles and potential vertical speed outputs for the AHRS. The groundspeed output option requires input of DME Distance (Label 202) and VOR Bearing (Label 222) over four additional 429 buses, one from each of two ARINC 711 VOR Receivers and two ARINC 709 DME Receivers.

Outputs shall be identical digital data (table 5-16) over three ARINC 429 buses operating at 100 kilobits per second.

TABLE 5-14. IRS DIGITAL SUMMARY - INPUTS/OUTPUTS (Reference 51)

OUTPUTS											
PARAMETER	OCTAL LABEL	SIGNAL FORMAT	MAXIMUM FILTER BANDWIDTH (HZ)	MAXIMUM TRANSPORT DELAY (NSEC)	MINIMUM UPDATE RATE (SPS)	SIGNIFICANT BITS/FIGURES	BINARY RANGE	APPROX RESOLUTION	ACCURACY	UNITS	POSITIVE SCALAR
ALONG TR. ACCEL	362	BNR	8*	60	50	12	±6	.001	10%	G	FORWARD
BODY LATERAL ACCEL	332	BNR	8*	60	50	12	±6	.001	10%	G	RIGHT
BODY LONGIT ACCEL	331	BNR	8*	60	50	12	±6	.001	10%	G	FORWARD
BODY NORMAL ACCEL	333	BNR	8*	60	50	12	±6	.001	10%	G	UP
BODY PITCH RATE	326	BNR	8*	50	50	13	±128	.001	10% 1 OR 1%	DEG/SEC	UP
BODY ROLL RATE	327	BNR	8*	50	50	13	±128	.001	10% 1 OR 1%	DEG/SEC	RIGHT WING DOWN
BODY YAW RATE	330	BNR	8*	50	50	13	±128	.001	10% 1 OR 1%	DEG/SEC	NOSE RIGHT
CROSS TR. ACCEL	363	BNR	8*	60	50	12	±6	.001	10%	G	RIGHT
DRIFT ANGLE	321	BNR	2	110	20	11	±18.75°	.09	23	DEG	RIGHT (L)
E & VELOCITY	367	BNR	2	110	10	13	±20%	.125	212	KNOTS	EAST
FLIGHT PATH ACCEL	323	BNR	8*	60	50	12	±6	.001	10%	G	FORWARD
FLIGHT PATH ANGLE	322	BNR	2	110	20	11	±180°	.09	24	DEG	UP
GROUND SPEED	312	BNR	2	110	10	13	0-2000	.125	212	KNOTS	ALWAYS POSITIVE
GROUND SPEED D	012	BCD	2		2	6	0-2000	1	23	KNOTS	ALWAYS POSITIVE
INERTIAL ALTITUDE	361	BNR	8*	60	25	18	±131,072	1	23	FEET	UP
INERTIAL VERT SPD	365	BNR	8*	60	25	18	±13,744	1	30	FT/MIN	UP
INSTRUMENTS	270	DIS		500	2	18	N/A	N/A		N/A	N/A
IRS MAINT. RATE	190	DIS			2	18	N/A	N/A		N/A	N/A
IRS TEST	277	DIS			2	18	N/A	N/A		N/A	N/A
MAGNETIC HEADING	320	BNR	2	110	20	13	±180	.0015	23	DEG	CW FROM NORTH
MAGNETIC HEADING D	018	BCD	2		2	6	0-359.9	.1		DEG	CW FROM NORTH
MAGNETIC HEADING D	166	BNR	2	110	10	13	±180°	.125	212	KNOTS	NORTH
PITCH ANGLE	324	BNR	8*	50	50	18	±180°	.011	7	DEG	UP
PITCH RATE	336	BNR	8*	50	50	13	±128	.015	1 OR 1%	DEG/SEC	UP
PLATTITUDE HEADING	336	BNR	2	110	10	11	±180	.01	24	DEG	CW FROM ZERO DEG
PRESENT POS LAT D	013	BCD		2	5	9	90S-90N	.1		DEG/MIN	NORTH FROM ZERO DEG
PRESENT POS LONG D	011	BCD		2	6	180E-180W	.1			DEG/MIN	EAST FROM ZERO DEG
PRESENT POS LAT	510	BNR	2	160	5	20	±180°	.000172		DEG	NORTH FROM ZERO DEG
PRESENT POS LONG	511	BNR	2	160	5	20	±180	.000172		DEG	EAST FROM ZERO DEG
ROLL ANGLE	325	BNR	8*	50	50	19	±180	.01	24	DEG	RIGHT WING DOWN
ROLL RATE	337	BNR	8*	50	50	13	±128	.015	1 OR 1%	DEG/SEC	RIGHT WING DOWN
TRACER ANGLE RATE	335	BNR	8*	50	50	11	±32	.015	23	DEG/SEC	CW
TRACER ANGLE TRUE	313	BNR	2	110	20	13	±180	.0015	23	DEG	CW FROM NORTH
TRACER ANGLE TRUE D	011	BCD		2	6	0-359.9	.1			DEG	CW FROM NORTH
TRACER ANGLE MAG	317	BNR	2	110	20	13	±180	.0015	23	DEG	CW FROM NORTH
TRUE HEADING	316	BNR	2	110	20	13	±180	.0015	23	DEG	CW FROM NORTH
TRUE HEADING D	004	BCD		2	6	0-359.9	.1			DEG	CW FROM NORTH
VERTICAL ACCEL	364	BNR	8*	60	50	12	±6	.001	10%	G	UP
WIND DIRECTION TRUE	316	BNR	2	110	10	8	±180	.7	218	DEG	CW FROM NORTH
WIND DIRECTION TRUE D	016	BCD		2	6	0-359.9	.1			DEG	CW FROM NORTH
WIND SPEED	313	BNR	2	110	10	8	0-256	1	212	KNOTS	ALWAYS POSITIVE
WIND SPEED D	015	BCD		2	5	0-256	1			KNOTS	ALWAYS POSITIVE
POTENTIAL VERT SPD	360	BNR	8*	60	25	18	±13,744	1	30	FT/MIN	UP

① When aircraft is resting on the ground, output is to be zero.

* SECOND ORDER BUTTERWORTH CHARACTERISTICS OR EQUIVALENT REQUIRED

0 OPERATING RANGE .90 DEG

Δ OPERATING RANGE .45 DEG

0-3 Accuracy specification constant altitude input and filter at steady state with no error assumed in air data input.

NOTE: Where differences appear between parameters characteristics listed in this document and ARINC Specification 709 (10/75), the latter should govern.

TABLE 5-14. IRS DIGITAL SUMMARY - INPUTS/OUTPUTS
(Reference 51) (Continued)

LABEL (OCTAL)	IRS DIGITAL	DATA			INPUTS		BCD		NOTES	
	PARAMETER	MAX	SIGNAL		RANGE (Scale)	SIG FIG	PAD FIG	UNITS		RESOL
	NAME	TRANSMIT INTERVAL	I/O							
			msec	In						
041	Set Latitude	500	X		90S-90N	5	0	Deg/Min	0.1	See Section 4.2.1
042	Set Longitude	500	X		180E-180W	6	0	Deg/Min	0.1	
043	Set Magnetic Hdg.	500	X		0-359.9	4	1	Deg	0.1	
203	Altitude	62.5	X		131,072	17	0	Feet	1	
210	True Airspeed	125	X		2,047.93	15		Knots	0.0625	

NOTE: Where differences appear between parameters characteristics listed in this document and ARINC Specification 429 (DITS), the latter should govern.

NOTES TO TABLE 5-14.

The signals on the IRS digital output bus will be used by other subsystems for various functions including display, navigation and closed loop control. Maximum pre-sampling filter bandwidth, maximum signal transport delay and minimum update rates are specified for output signals according to their anticipated use in the system. Selection of these three interrelated parameters is based on considerations of data freshness, signal and noise aliasing, and control loop stability.

Note 1: Maximum Filter Bandwidth

The maximum 3 dB bandwidth for the pre-sampling (anti-aliasing) filter applicable to each parameter is specified in the fourth column. A filter equivalent to a first order lag is required except where a second order Butterworth characteristic is required as denoted by an asterisk. The minimum bandwidth is constrained by the transport delay requirement. The filtering must be performed before sampling or making computations at the specified update rate.

The maximum filter bandwidth requirement refers to overall filtering of the parameter. The filtering may be a combination of hardware filters (shock mounts, analog, or digital) or software digital filters. The conversion of the sensed parameter to digital form may also limit the bandwidth.

Parameters derived from sensed filtered parameters may also have a stated maximum bandwidth. If the bandwidth of the derived parameter exceeds the specified maximum bandwidth, then further filtering is required.

Note 2: Maximum Transport Delay

The maximum allowable transport delay for each parameter is defined as the total time delay in the signal path for a parameter from the sensor input(s) to the output data bus. Effects of all sources of delay are included in this quantity (e.g., sensor and filter phase delay, computation delay and output delay). When a parameter is output from a complementary filter or similar process where wideband and narrowband data are combined, the transport delay requirement applies to the wideband portion of the process.

Note 3: Minimum Update Rate

The minimum allowable update rate for each parameter is specified. Significantly higher rates, although technically acceptable, are discouraged because of potentially excessive input burden on the using computers. Fresh data should be calculated and transmitted at the update rate.

TABLE 5-15. DIGITAL CONTROL WORD FORMAT IRS DISCRETE WORD FORMAT

Label Code 270 04
Minimum Transmit Interval: 500 msec

Bit No.	Function	Bit Status	
		1	0
1	Label	x	
2			x
3		x	
4		x	
5		x	
6	SDI*		x
7			x
8			x
9			
10			
11	Align Mode/Not Ready	Yes	No
12	Reversionary Attitude Mode	Yes	No
13	Normal Mode	Yes	No
14	Set Heading	Yes	No
15	Attitude Invalid	Yes	No
16	DC Fail (Low)	Yes	No
17	ON DC	Yes	No
18	ADC Fault	Yes	No
19	IRU Fault	Yes	No
20	DC Fail - ON DC	Yes	No
21	Align Fault	Yes	No
22	No IRS Initialization	Yes	No
23	EXCESSIVE MOTION ERROR	Yes	No
24	ADC/IRU Fault	Yes	No
25	No VOR/DME No. 1 Input	Yes	No
26	Align Status		
27			
28			
29		Yes	No
30			
31	SSM*		
32	Parity (Odd)		

*See ARINC Specification 429 for information concerning the SDI and SSM fields.

NOTES TO TABLE 5-15

1. The above digital format assignment is a more complete definition of the IRS Discrete Control Word Label 270 04 assigned in ARINC 429 DITS - Attachment 2 table 3.1.

2. More complete definition of discrete bit conditions are as follows:

Bit 11. Align Mode/Not Ready: The IRU operating software mode is ALIGN or the initialization of any mode.

Bit 12. Reversionary Attitude Mode: The IRU operating software mode is ATT.

Bit 13. NAV Mode: The IRU operating software mode is NAV.

Bit 14. Set Heading: Magnetic heading outputs are no longer being calculated but have the characteristics of a "free DG" and a set heading has been input to the IRU. (See Section 3.2.4 of characteristic for further explanation.)

Bit 15. Attitude Invalid: The IRU has detected a failure of attitude, heading, angular body rates, or linear body accelerations (same as FAULT discrete).

Bit 16. DC Fail: The IRU DC power input is less than 18 VDC.

Bit 17. On DC: The IRU is operating on the DC power input.

Bit 18. ADC Fault: ADC in-flight fault, but power-on BITE found no faults with the IRU ADC input channel.

Bit 19. IRU Fault: The BITE has detected a fault not annunciated in Bits 18, 21, 22, 23, or 24.

Bit 20. DC Fail - On DC: The DC power input was not available when required by the IRU. This condition shall be reset only by power-on initialization.

Bit 21. Align Fault: Failed the IRU operating software ALIGN criterion but neither power-on nor continuous BITE show any faults.

Bit 22. No IRS Initialization: No input or an incorrect input has been received from the IRMP or FMC's.

Bit 23. EXCESSIVE MOTION ERROR: Non-zero groundspeed during the ALIGN mode.

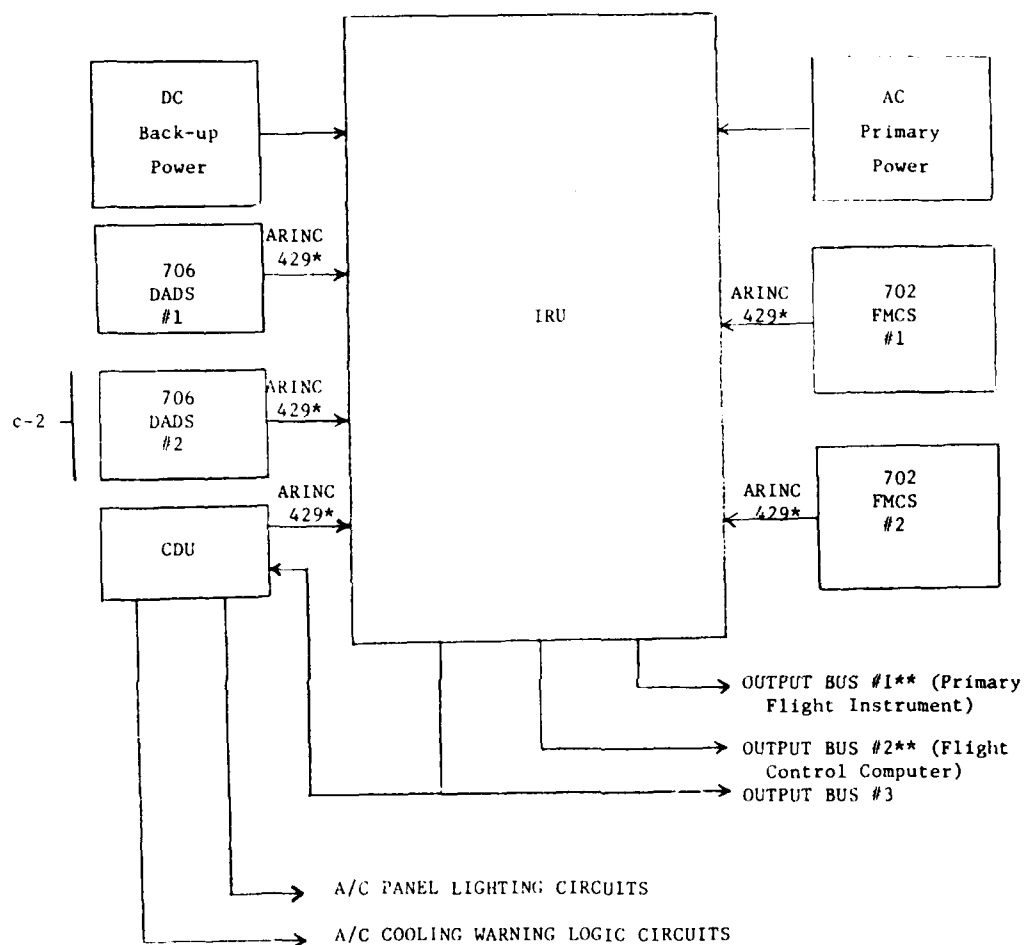
Bit 24. ADC/IRU Fault: ADC in-flight fault, but no power-on BITE information available prior to next flight.

Bit 25. No VOR/DME No. 1 input.

Bit 26, 27, 28. Align status is represented by a series of descending digits, each indicating a successive state of alignment. Three bits provide a seven state alignment status as follows:

	LSB		MSB	
Bit Number	26	27	28	
	1	1	1	Alignment Commenced
	0	1	1	-
	-	-	-	-
	-	-	-	-
	-	-	-	-
	1	0	0	Highest Alignment Status
	0	0	0	Unassigned

Bit 29. No VOR/DME No. 2 input.



*12-14.5 KHz
 **100 KHz

FIGURE 5-19. 704 BLOCK DIAGRAM

TABLE 5-16. AHRS DIGITAL OUTPUT SUMMARY

PARAMETER	OCTAL LABEL	SIGNAL FORMAT	MAXIMUM FILTER BANDWIDTH (HZ)	MAXIMUM TRANSPORT DELAY (MSEC)	MINIMUM UPDATE RATE (SPS)	SIGNIFICANT BITS/FIGURES	BINARY RANGE	APPROX RESOLUTION	ACCURACY	UNITS	POSITIVE SENSE	SELF TEST VALUE
ALONG HDG ACCEL	373	BNR	8*	60	30	12	±4	.001	10%	G	FORWARD	.02G
BODY LATERAL ACCEL	332	BNR	8*	60	30	12	±4	.001	±.01	G	RIGHT	.10G
BODY LONGIT ACCEL	331	BNR	8*	60	30	12	±4	.001	±.01	G	FORWARD	.02G
① BODY NORMAL ACCEL	333	BNR	8*	60	30	12	±4	.001	±.01	G	UP	.10G
BODY PITCH RATE	326	BNR	8*	50	30	13	±128	.013	.1 OR 1%	DEG/SEC	UP	10°/SEC
BODY ROLL RATE	327	BNR	8*	50	30	13	±128	.013	.1 OR 1%	DEG/SEC	RIGHT WING DOWN	10°/SEC
BODY YAW RATE	330	BNR	8*	50	30	13	±128	.013	.1 OR 1%	DEG/SEC	NOSE RIGHT	10°/SEC
CROSS HDG ACCEL	376	BNR	8*	60	30	12	±4	.001	10%	G	RIGHT	.02G
DRIFT ANGLE	321	BNR	2	110	10	11	±180°	.09	.5	DEG	RIGHT	-10° (LI)
E-W VELOCITY-MAG	370	BNR	2	110	10	13	±4096	.123	±12	KNOTS	EAST	200 KNOTS (L)
FLIGHT PATH ACCEL	323	BNR	8*	60	30	12	±4	.001	10%	G	FORWARD	.02G
FLIGHT PATH ANGLE	322	BNR	2	110	20	11	±180 Δ	.09	.5	DEG	UP	.5°
GROUND SPEED	312	BNR	2	110	10	13	0-4096	.123	±12	KNOTS	ALWAYS POSITIVE	200 KNOTS
GROUND SPEED-D	012	BCD			2	6	0-2000	1		KNOTS	ALWAYS POSITIVE	200 KNOTS
INERTIAL ALTITUDE	361	BNR	8*	63	23	18	±131,072 □	.5	±5 ∇	FEET	UP	10,000 FT
INERTIAL VERT SPD	363	BNR	8*	63	23	15	±32,768	1	30	FT/MIN	UP	-400 FT/MIN
AHRS DISCRETE	270	DIS		300	2	19	N/A	N/A		N/A	N/A	N/A
AHRS MAINT DISCRETE	330	DIS			2	18	N/A	N/A		N/A	N/A	N/A
AHRS TEST	277	DIS			2	18	N/A	N/A		N/A	N/A	N/A
MAGNETIC HEADING	320	BNR	2	110	20	13	±180	.0033	±2	DEG	CW FROM NORTH	15°
MAGNETIC HEADING-D	010	BCD			2	6	0-359.9	.1		DEG	CW FROM NORTH	15°
N-S VELOCITY-MAG	373	BNR	2	110	10	13	±4096	.123	±12	KNOTS	UP	200 KNOTS (L)
PITCH ANGLE	326	BNR	8	50	30	16	±180	.011	.5	DEG	UP	.5°
PITCH ATT RATE	334	BNR	8*	50	30	13	±128	.013	.1 OR 1%	DEG/SEC	UP	10°/SEC
PLATFORM HEADING	330	BNR	2	110	10	11	±180	.09	.5	DEG/HR	CW FROM ZERO DEG	22.50°
ROLL ANGLE	323	BNR	8	50	30	16	±180	.01	.5	DEG	RIGHT WING DOWN	.5° (R)
ROLL ATT RATE	337	BNR	8*	50	30	13	±128	.013	.1 OR 1%	DEG/SEC	UP	10°/SEC
TRACK ANGLE RATE	335	BNR	8*	40	23	11	±32	.013	.23	DEG/SEC	CW FROM NORTH	.4°/SEC
TRACK ANGLE-MAG	317	BNR	2	110	20	13	±180	.0033	±6	DEG	CW FROM NORTH	5°
TRACK ANGLE-MAG-D	033	BCD			2	6	0-359.9	.1		DEG	CW FROM NORTH	10°
① VERTICAL ACCEL	360	BNR	8*	60	30	12	±4	.001	±.01	G	UP	0.1G
WIND DIRECT-MAG	372	BNR	2	110	10	8	±180	.7	±10	DEG	CW FROM NORTH	30°
WIND DIRECT-MAG-D	036	BCD			2	3	0-359.9	1		DEG	CW FROM NORTH	30°
WIND SPEED	313	BNR	2	110	10	8	0-256	1	±12	KNOTS	ALWAYS POSITIVE	100 KNOTS
WIND SPEED-D	013	BCD			2	3	0-256	1		KNOTS	ALWAYS POSITIVE	100 KNOTS
POTENTIAL VERT SPD	360	BNR	8*	63	23	13	±32,768	1	30	FT/MIN	UP	-400 FT/MIN

① When aircraft is resting on the ground, output is to be zero.

* SECOND ORDER BUTTERWORTH CHARACTERISTICS OR EQUIVALENT REQUIRED

Δ OPERATING RANGE ±90 DEG

□ OPERATING RANGE ±95 DEG

□ Uses the S01 for least significant bits.

∇ Accuracy specified with constant altitude input and filter at steady state with no error assumed in air data input.

NOTE: Where differences appear between parameter characteristics listed in this document and ARINC Specification 629 (DITS), the latter should prevail.

NOTES TO TABLE 5-16

The signals on the AHRS digital output bus will be used by other subsystems for various functions including display, navigation, and closed loop control. Maximum presampling filter bandwidth, maximum signal transport delay, and minimum update rates are specified for output signals according to their anticipated use in the system. Selection of these three interrelated parameters is based on considerations of data freshness, signal and noise aliasing, and control loop stability.

Note 1: Maximum Filter Bandwidth

The maximum 3 dB bandwidth for the presampling (anti-aliasing) filter applicable to each parameter is specified in the fourth column. A filter equivalent to a first order lag is required, except where a second order Butterworth characteristic is required as denoted by an asterisk. The minimum bandwidth is constrained by the transport delay requirement. The filtering must be performed before sampling or making computations at the specified update rate.

The maximum filter bandwidth requirement refers to overall filtering of the parameter. The filtering may be a combination of hardware filters (shock mounts, analog, or digital) or software digital filters. The conversion of the sensed parameter to digital form may also limit the bandwidth.

Parameters derived from sensed filtered parameters may also have a stated maximum bandwidth. If the bandwidth of the derived parameter exceeds the specified maximum bandwidth, then further filtering is required.

Note 2: Maximum Transport Delay

The maximum allowable transport delay for each parameter is defined as the total time delay in the signal path for a parameter from the sensor input(s) to the output data bus. Effects of all sources of delay are included in this quantity (e.g., sensor and filter phase delay, computation delay, and output delay). When a parameter is output from a complementary filter or similar process where wideband and narrowband data are combined, the transport delay requirement applies to the wideband portion of the process.

Note 3: Minimum Update Rate

The minimum allowable update rate for each parameter is specified. Significantly higher rates, although technically acceptable, are discouraged because of potentially excessive input burden on the using computers. Fresh data should be calculated and transmitted at the update rate.

5.6.7 Air Data System.

ARINC Characteristic 706-2 (Reference 53) describes a subsonic digital air data system (DADS) which outputs the data in table 5-17 over four ARINC 429 low speed buses. The two discrete output word formats are given in tables 5-18 and 5-19.

The DADS also has two ARINC 429 input ports for input of baro-correction information in BCD format. This information is used from both ports only if the baro-correction program pin and baro-correction input discrete pin are connected to program pin common. If the baro-correction input discrete pin is open circuit, only port "A" (digital) or the #1 input (analog) is used, depending on the state of the baro-correction program pin (open-use analog). Additional discussion of input discretes is in reference 53.

5.6.8 Radio Altimeter.

ARINC Characteristic 707-3 (reference 54) describes the functions of the radio altimeter which outputs radio height in BNR and BCD codes (Labels 164 and 165, respectively) as well as the radio altimeter check point deviation (Label 166) over two ARINC 429 low speed digital data buses. Bus no. 1 output is intended for the exclusive use of the Automatic Flight Control System (AFCS) while bus No. 2 output is intended for use by the displays and all other systems, including secondary inputs to the AFCS (reference 54).

5.6.9 Airborne Weather Radar.

ARINC Characteristic 708-2 (reference 55) describes the architecture of digital weather radar system configurations. All of the configurations accept stabilization signals which conform to ARINC 704 or ARINC 705 serial digital outputs. Serial digital control data is input from up to four ARINC 429 data buses. The output data to the display is transmitted on two special versions of the ARINC 453 Very High Speed Serial Digital bus. A third output bus is optional. The format of the 32-bit control word for the radar is given in reference 55.

TABLE 5-17. AIR DATA SYSTEM DIGITAL
DATA OUTPUT STANDARDS
(Reference 53)

This table summarizes, for reference purposes only, the digital data standards applicable to the ARINC 706 ADS. Note that, as stated in Section 4.9.3 of this document, ARINC Specification 429, "Mark 33 Digital Information Transfer System (DITS)" is the controlling document for these standards. In the event of a conflict between the material in this table and Specification 429, the latter should be assumed to be correct.

Parameter	Label (Octal)		Computer Operational Range	Maximum Transmit Interval (msec)	Binary Output			BCD Output		
	BNR	BCD			BNR Word Range	Signif. Bits	Resolution & Units	Signif. Figures	Least Signif. Fig.	Maximum Transmit Interval (msec)
Altitude (1013.25 mb)	203		-1000-+50,000 ft.	62.5	131071	17	1 Foot			
Baro Corrected Altitude Output #1	204		-1000-+50,000 ft.	62.5	131071	17	1 Foot			
Baro Corrected Altitude Output #2	220		-1000-+50,000 ft.	62.5	131071	17	1 Foot			
Altitude Rate	212		0-+20,000ft/min.	62.5	32752	11	16Ft/min.			
Computed Airspeed	206		30-450 Knots	125	1023.75	14	0.0625 Knot			c-2
Max. Allowable Airspeed	207		150-450 Knots	125	1023.75	12	0.25 Knot			
Mach	205		0.10-1.00 Mach	125	4.096	16	0.0000625 Knot			c-2
True Airspeed	210	230	100-599 Knots	125	2047.93	15	0.0625 Knot	3	1 Knot	500
Static Air Temp	213	233	-99-+60°C	500	511.75	11	0.25°C	2	1°C	500
Total Air Temp	211	231	-60-+99°C	500	511.75	11	0.25°C	2	1°C	500
Impact Pressure	215	-	0-372.5 mb	125	511.97	14	0.03125 mb			c-2
Total Pressure	242		135.5-1354.5 mb	125	2047.97	16	0.03125 mb			
Baro Correction mb#1		234	745-1050mb					5	0.1 mb	125
Baro Correction Ins. Hg #1		235	22.00-31.00In Hg					5	0.001InHg	125
Baro Correction mb#2		236	745-1050mb					5	0.1 mb	125
Baro Correction Ins. Hg #2		237	22.00-31.00In Hg					5	0.001InHg	125
Indicated Angle of Attack	221		-60°-+60°	62.5	±180°	12	0.05°			
Corrected Angle of Attack	241		-60°-+60°	62.5	±180°	12	0.05°			
Discrete Word #1	270		see 4.10.17	500						c-2
Discrete Word #2	271		see 4.10.17	500						
Maintenance Word #1	350		see 4.6	500						
Maintenance Word #2	351		see 4.6	500						

TABLE 5-18. ADS DISCRETE WORD #1 FORMAT (Reference 53)

Label Code 270 (Octal)
Maximum Transmit Interval: 500 msec

Bit No.	Function	Bit Status	
		1	0
1	Label	X	
2			X
3		X	
4		X	
5		X	
6	SDI*		X
7			X
8			X
9			
10			
11	Iceing Detector	On	Off
12	Pitot Probe Heat	On	Off
13	ADS Computer Status	Fail	Good
14	Pitot/Static Probe Heat	On	Off
15	Static Source Heat	On	Off
16	TAT Probe Heat	On	Off
17	Left Side Angle of Attack Sensor Heat	On	Off
18	Right Side Angle of Attack Sensor Heat	On	Off
19	VMO/MMO Overspeed Warning	Warn	Not Warn
20	Primary Angle of Attack Input	Fail	Good
21	Angle of Attack Average	Yes	No
22	VMO Alternate No. 1	Yes	No
23	VMO Alternate No. 2	Yes	No
24	VMO Alternate No. 3	Yes	No
25	VMO Alternate No. 4	Yes	No
26	SSEC Alternate	Yes	No
27	Angle of Attack Alternate Correction	Yes	No
28	Baro-Correction Port "A"	Yes	No
29	Zero Mach SSEC	Yes	No
30	SSM*		
31			
32	Parity (Odd)		

*See ARINC Specification 429 for information concerning the SDI and SSM fields.

TABLE 5-19. ADS DISCRETE WORD #2 FORMAT (Reference 53)

Label Code 271 (Octal)
Maximum Transmit Interval: 500 mSec

Bit No.	Function	Bit Status	
		1	0
1	Label		
2			
3			
4			
5			
6			
7			
8	SDI*	Yes	No
9			
10			
11	Zero Angle of Attack SSEC	Fail	Good
12			
13	Angle of Attack Sensor Status		
14			
15			
16			
17			
18			
19			
20			
21			
22			
23	Spare		
24			
25			
26			
27			
28			
29			
30	SSM*		
31			
32	Parity (Odd)		

c-2

*See ARINC Specification 429 for information concerning the SDI and SSM fields.

5.6.10 Airborne Distance Measuring Equipment.

ARINC Characteristic 709-4 (reference 56) describes the desired design characteristics of a new digital DME which provides slant range distance from the aircraft to a selected DME ground facility. The DME has two serial digital data input ports. "The mode of operation of the DME will be determined by the state of bit numbers 11 through 17 in the frequency functions words" as defined in table 5-20 (reference 56).

TABLE 5-20. 709 DME MODE SELECTION MATRIX

IDENT	DISP	MLS	ILS	DME FUNCTION			INTERPRETATION
17	16	15	14	13	12	11	BIT NO.
X	X	X	X	0	0	0	STANDBY
				0	0	1	DIR FREQ 1
				0	1	0	DIR FREQ 2
				0	1	1	DIR FREQ 3
				1	0	0	DIR FREQ 4
				1	0	1	DIR FREQ 5
X	X	X	X	1	1	0	FREE SCAN X
X	X	X	X	1	1	1	FREE SCAN X & Y

"Bit No. 14 will be used for ILS paired frequencies and Bit No. 15 will be used for MLS paired frequencies. A binary "one" in Bit No. 14 indicates an ILS paired frequency. A binary "one" in Bit No. 15 indicates an MLS paired frequency. Both bits high (binary "one") would be unused.

COMMENTARY

"Single Channel (manual) mode for display with audio IDENT is selected by setting bit nos. 13, 12, and 11 to 'zero,' 'zero' and 'one,' and bit nos. 17 and 16 to 'one' and 'one'" (reference 56).

The 709 DME outputs identical streams of time-multiplexed BCD-encoded DME channel frequency (Label Code 035) data followed by BNR (Label Code 202) and BCD-encoded distance (Label Code 201) data from two identical low speed ARINC 429 serial ports at a rate of six words per second minimum. Additional information on functional test, monitoring, and the frequency scanning mode are given in reference 56.

5.6.11 ILS Receiver.

ARINC Characteristic 710-3 (reference 57) describes the desired characteristics of an airborne ILS receiver that can be tuned through the serial digital frequency/function selection system described in ARINC Specification 720. Two serial digital data input ports are available for input of the frequency and runway heading information entered by the pilot. The outputs from the ARINC 710 ILS receiver are identical streams of time multiplexed BNR-encoded localizer and glide slope deviation data and BCD-encoded ILS channel frequency data transmitted through two

identical ARINC 429 ports at the "low" bit rate. The "highest" integrity port is designated for the automatic flight control system, and the second port is for other utilization devices such as the EFIS. Each localizer deviation word and each glide slope deviation word should be repeated not less than 16 times per second. The ILS channel frequency word and runway heading word repetition rates should be the same as that of the incoming data. The ISO Alphabet No. 5-encoded ILS facility identifier information, if supplied to the receiver, should be transmitted at the same rate as the incoming message (nominally five times per second). Additional information is contained in reference 57.

5.6.12 Airborne VOR Receiver.

ARINC Characteristic 711-4 (reference 58) describes the desired characteristics of a VOR receiver designed to utilize the serial digital frequency/function selection system described in ARINC Specification 720. Two serial digital data input ports are provided. In some system architecture using ARINC 711 VOR's, the source of frequency information will also be the source of selected course information entered by the pilot. The marker beacon receiver element of the ARINC 711 VOR receiver provides outputs encoded in bits 11, 12, and 13 of the omnibearing output word for indicating the passage of the aircraft over the marker beacon components of the instrument landing system (ILS).

The VOR receiver has two identical ARINC 429 output ports which operate at the "low" bit rate. Identical streams of time-multiplexed BNR-encoded omnibearing data (Label Code 222) and BCD-encoded VOR channel frequency data (Label Code 034) are output at rates of not less than 16 times per second and five times per second, respectively. In system architectures which use the selected course words, the input words are multiplexed onto the output buses at repetition rates the same as those of the incoming data. Additional information is contained in reference 58.

5.6.13 Airborne ADF System.

ARINC Characteristic 712-3 (reference 59) describes the desired characteristics of an ADF receiver designed to utilize the serial ARINC 720 Digital Frequency/Function Selection System (DFFSS) described in ARINC specification 720 via either of two digital input ports. Outputs from the ARINC 712 ADF should be identical streams of time-multiplexed BNR-encoded omnibearing data (Label Code 162 (octal)) from two identical but mutually isolated ports at the "low" bit rate (12 to 14.5 kilobits per second). Reference 59 presents examples of the BNR ADF bearing data coding.

5.6.14 Mark 2 Omega Navigation System.

ARINC Characteristic 599 (reference 60) sets forth the characteristics of an airborne "stand alone" OMEGA Navigation System which is capable of accepting digital inputs through ARINC 429 ports from a second OMEGA system, the 706 DADS, the 702 FMCS, and either the 704 IRS or the 705 AHRS. The system is to provide digital output through ARINC 429 ports (five total). ARINC may, in the future, update this characteristic to reflect the latest modifications to Characteristic 429.

5.6.15 Airborne VHF Communications Transceiver.

ARINC Characteristic 716-3 (reference 61) describes the desired characteristic of a VHF transceiver designed to utilize the serial digital frequency/function selection system via either of two serial digital data input ports. Further information is contained in reference 61.

5.6.16 Flight Data Acquisition and Recording System.

ARINC Characteristic 717-3 (reference 62) describes a Digital Expandable Flight Data Acquisition (DEFDAU) unit and Recording System which, while utilizing ARINC 429 buses for input and output of digital data, must record data in the same format as described in ARINC Characteristic 573 (reference 63). There are 16 separate ARINC 429 ports in the basic DEFDAU. Provision has been made in the connector for an additional eight ARINC 429 input ports. An ARINC 429 digital output port can be used as an option. Additional information such as the assignment of specific signals to connector pins is contained in reference 62.

5.6.17 Mark 3 Air Traffic Control Transponder (ATCRBS/DABS).

ARINC Characteristic 718-3 (reference 64) describes an Air Traffic Control Radar Beacon System (ATCRBS) transponder which is controlled through two serial digital data ARINC 429 input ports using the ARINC 720 DFFSS. The system also has two ports for input from the 706 DADS, two ports for input from either the 704 IRS or 702 FMC, a single crew terminal input port, and two spare input ports. There are two ARINC 429 digital output ports, one for the crew terminal and one for the DABS. Additional information on message content is given in reference 64.

5.6.18 Digital Frequency/Function Selection for Airborne Electronic Equipment.

ARINC Characteristic 720-1 (reference 65) describes the characteristics of a Digital Frequency/Function Selection System (DFFSS) to be used for selection of frequency/function of the receivers on the aircraft. Each radio is to be provided with two ARINC 429 low speed (12-14.5 KHz) input ports and one input port selection discrete. The DFFSS has two possible architectures; a centralized system using two identical subsystems and a federated configuration using a dedicated control for each radio. Provisions are made in the dedicated control units to receive remotely selected frequencies.

Each centralized DFFSS subsystem has two digital data output ports for radio or other electronic system control signals, two digital data input ports for frequency selection information generated in a remote source such as a flight management computer, and one input and one output port for DFFSS subsystem to DFFSS subsystem transfer of digital data.

Each subsystem should be able to drive five radio/electronic systems without deterioration of system performance. Additional information on applications is given in reference 65.

5.6.19 Ground Proximity Warning System.

ARINC Characteristic 723-1 (reference 66) describes a ground proximity warning system (GPWS) which provides audible and visible warnings or alerts when an aircraft approaches terrain more closely than, or deviates downward from an ILS glide slope beyond, the limits set into the system. Four ARINC 429 serial digital data buses provide inputs from the 706 DADS, 707 Radio Altimeter, 710 ILS receiver, and either the 704 IRS or the 705 AHRS. A 429 serial digital output port provides data to the caution advisory computer. A description of the discrete output word format is given in table 5-21.

TABLE 5-21. DISCRETE WORD FORMAT (Reference 66)

GPWS Discrete - Label 270 23

Bit No.	Function	Bit Status	
		1	0
1	Label	x	
2			x
3		x	
4		x	
5		x	
6			x
7			x
8			x
9	SDI		
10			
11			
12			
13	Sink Rate		
14	Pull Up		
15	Terrain		
16	Don't Sink		
17	Too Low Gear	*	
18	Too Low Flap		
19	Too Low Terrain		
20	Glide Slope		
21	Minimum Minimum		
22	Terrain Pull Up		
23	Spare (All "0" states)		
24			
25			
26			
27			
28	SSM		
29			
30			
31	Parity (Odd)		
32			

Source: Maintenance Output Port

*Only one visual message should be displayed at a time (only one data bit should be set to the logic "1" state at a time).

5.6.20 Electronic Flight Instruments (EFI).

ARINC Characteristic 725-1 (reference 67) describes the architecture of an EFI system made up of one or more each of the following components:

- (1) Symbol Generator/Processor Unit (SG).
- (2) Cockpit located Control Panel (CP).
- (3) Raster and/or Stroke Written Multicolor Cathode Ray Tube Display.

The EFI system is designed to provide all the conventional Attitude Director Indicator (ADI) and Horizontal Situation Indicator (HSI) functions in addition to air data functions, map display, weather data, radio altitude information, automatic flight control mode annunciation, flightpath information, and flight warning display. "Display of the basic flight essential ADI, HSI, and air data functions should not be dependent upon proper operation of any sensors or subsystems being operational except the sensor that provides the essential data. The map display function should be generated based upon preprocessed data from an ARINC 702 Flight Management Computer System" (reference 67).

Table 5-22 summarizes the digital inputs to the signal generator. Note the color weather radar has two ARINC 453 1 MHz serial digital outputs which are inputs to the signal generator. The signal generator has an ARINC 429 serial digital output port for use with the maintenance monitoring system. The 725 control panel has two serial digital output ports for interface to the Flight Management Computer(s). System design considerations concerning the display units, control panel selectable modes, and data source switching are contained in reference 67.

TABLE 5-22. ARINC 725-1 SIGNAL GENERATOR DIGITAL INPUTS
(Reference 67)

<u>Number</u>	<u>Type</u>	<u>Inputs From</u>	<u>Input Port Speed</u>
2	2	725 Control Panel	Low
2		703 Thrust Control Computer	Low
2	1	702 Flight Management Computer	High
3		701 Flight Control Computer	Low
3	4	704 Inertial Reference System	High
3	4	710 Instrument Landing System	Low
2	1	709 Distance Measuring Equipment	Low
2	1	711 VHF Omirange (VOR)	Low
3	4	707 Low Range Radio Altimeter	Low
2		708 Weather Radar	453
2	1	706 Air Data System	Low
2	1	712 Automatic Direction Finder	Low
3	4	727 Microwave Landing System	Low
2	1	Flight Augmentation Computer	Low
2		726 Flight Warning Computer	High
2		Glare Shield Controller	Low

5.6.21 Flight Warning Computer System.

Table 5-23 lists the serial digital inputs to the FWCS. The system has four serial digital outputs: (1) To the second FWCU; (2) to the Warning Caution Display; and (3) to the AIDS/Maintenance. Additional information on system design considerations is given in reference 68.

TABLE 5-23. ARINC 726-1 FLIGHT WARNING COMPUTER SYSTEM
DIGITAL INPUTS (Reference 68)

<u>Number</u>	<u>Type</u>	<u>Inputs From</u>	<u>Input Port Speed</u>
2		707 Radio Altimeters	Low
2		706 Digital Air Data System	Low
3		704 IRS or 705 AHRS	High
2		710 Instrument Landing System	Low
2		Engine/Fuel/Airframe	Low
1		Second 726-1 FWCU	Low
1		Warning and Caution Display	Low
3		701 Flight Control Computer	
1		703 Thrust Control Computer	
2		Spare	Low

5.6.22 Airborne MLS Receiver.

ARINC Characteristic 727, Part 1, Aircraft Installation Provisions (reference 69) describes design considerations for a microwave landing system receiver. Frequency and azimuth radial selection are input through the two ARINC 429 serial digital input ports. Six serial digital output ports are provided for interface with the AFCS and warning devices. Six serial digital input ports are provided for interface to data devices. Further design considerations are described in reference 69.

5.6.23 Analog and Discrete Data Converter System.

ARINC Characteristic 729-1 (reference 70) describes a system intended to process, convert, and multiplex analog and discrete signals in order to provide them in the ARINC 429 format.

The system has two identical ARINC 429 high-speed digital output ports. The reference describes alternative architectures and input signal characteristics.

5.6.24 Airborne Separation Assurance System.

ARINC Characteristic 730-3 (reference 71) describes the airborne elements of the Discrete Address Beacon System (DABS)/Air Traffic Control Radar Beacon System (ATCRBS) Air Traffic Control (ATC) surveillance and data link systems and an active Beacon-based Collision Avoidance System (BCAS). The BCAS computer and the DABS/ATCRBS transponder share a common ARINC Specification 600 package which is supplemented with a collision avoidance maneuver command and advisory display and system control panel.

The serial digital inputs to the system are listed in table 5-24. Note that one of the buses is the ARINC 453 bus operating at 1 MHz while the others are ARINC 429

buses. The system has two ARINC 429 output buses and one ARINC 453 output bus. Further data are contained in the reference data standards considered for the standard message (SM) and extended length message (ELM).

TABLE 5-24. ARINC 730-3 AIRBORNE SEPARATION ASSURANCE SYSTEM
DIGITAL INPUTS (Reference 71)

<u>Number</u>	<u>Type</u>	<u>Inputs From</u>	<u>Input Port Speed</u>
2		Transponder Control Unit	Low
1		Crew Terminal Interface #2	
2		706 Digital Air Data System	Low
1		707 Radio Altimeter	
2		704 Inertial Reference System or 705 AHRS	High
3		429 Spare Bus	
1		453 Bus	1 MHz

5.6.25 Electric Chronometer.

ARINC Characteristic 731-1 (reference 72) describes a self-contained electric chronometer which provide Greenwich Mean Time (GMT), month, and day data in digital format over a single ARINC 429 low speed bus to other digital avionics. Additional information on design information and accuracy is contained in reference 72.

5.6.26 Digital Engine Controller.

ARINC has not published a characteristic for such a system but various manufacturers are designing these systems (references 38 and 73) for control of the engines on the next generation of transport aircraft. Many of the designs have either ARINC 429 input and output ports or MIL-STD-1553B interfaces. Input data are from subsystems such as the ARINC 706 Digital Air Data System. Output data are engine parameters such as N1, EPR, etc. Most of the designs consist of dual channels with digital data exchange between channels. Reference 38 selected a serial interface between the dual channels. Since the engine controllers do not have an industry wide interface established, particular care must be taken in the validation process to determine if any faults can occur in the digital computation of the engine control parameters that could result in loss of thrust or engine control.

5.6.27 Electric Power Generation System.

ARINC has not published a characteristic for an electric power generation system utilizing digital control. Various aircraft manufacturers are designing electrical power distribution systems which utilize digital data buses to control electrical load management centers which interface with solid state power controllers. The digital data buses also interface with the generator control unit (references 74 and 75). As these concepts move from the laboratory to the future generations of aircraft, it is probable that new standards for their design will be agreed on by the developers and users.

5.6.28 Synopsis of Digital Interfaces of ARINC Characteristic Subsystems.

Table 5-25 presents a summary of the number of digital interfaces to the subsystems described by the relevant ARINC characteristics. The interwiring between subsystems is basically point-to-point. The user of the handbook should be sure to have the latest version of the ARINC characteristic available for up-to-date information on a specific subsystem since supplements often result in major changes to the previous versions of the characteristic.

TABLE 5-25. SYNOPSIS OF DIGITAL INTERFACES OF ARINC CHARACTERISTIC SUBSYSTEMS

ARINC Characteristic	Number	Inputs From	Speed	Digital		
				Number	Outputs To	Speed
701 FCCU	2	705/704 AHRS/IRS	HI		702 FMCS	
		706 DADS	LO		701 FCCU	
		710 ILS Revr.				
		707 Radio Alt.				
		702 FMCS				
		AFS Controller				
		FAC				
		703 TCCS				
		709 DME				
		INS/OMEGA				
		2nd 701 FCCU				
701 AFS Controller	2	Instrument Buses		3	701 FCCU	
	2	701 FCCU		1	General #1	
	1	701 FCCU or 703 TCC		1	General #2	
702 Flight Management Computer System	2	725 EFIS	LO	2	725 EFIS	HI
	2	711 VOR		1	General Output #2	LO
	2	709 DME		1	General Output #2	LO
	1	710 ILS		1	CDU Port A	
	2	706 DADS	LO	1	CDU Port B	
	3	704/705 IRS/AHRS	HI	1	603 Data Load	
	1	701 Controller				
	2	CDU				
	1	603 Data Load				
	1	724 ACARS		1	724 ACARS	LO
	1	702 2nd FMC		1	702 2nd FMC	

TABLE 5-25. SYNOPSIS OF DIGITAL INTERFACES OF ARINC
CHARACTERISTIC SUBSYSTEMS (Continued)

ARINC Characteristic	Number	Inputs From	Speed	Digital		
				Number	Outputs To	Speed
702 FMCS (Cont'd)	6	703 TCC Inc. 4 Engines				
	1	Clock				
	1	Omega				
	1	MLS				
	2	Sparcs				
703 TCCU	2	706 DADS	LO	2	General Output	
	2	704/705 IRS/AHRS	HI			
	1	707 Radio Altimeter				
	2	702 FMC				
	3	701 FCC				
	1	703 2nd TCCU				
	4	FADEC				
	1	701 Controller				
704 IRS	1	CDC	LO	1	725 EFIS	HI
	2	702 FMC	LO	1	FCCU	HI
	2	706 DADS	LO	1	CDU, etc.	HI
705 AHRS	2	706 DADS		1	725 EFIS	HI
	2	709 DME		1	701 FCCU	HI
	2	711 VOR		1	General Data	HI
706 DADS	2	Baro-Correction		1	AFS	LO
				1	725 EFIS	LO
				1	FADEC	LO
				1	General	LO
707 Radio Altimeter	None			1	AFS	LO
				1	Displays, etc.	LO
708 Airborne Weather Radar	1	704/705 IRS/AHRS	HI	3	ARINC 453	
	4	Control	LO			

TABLE 5-25. SYNOPSIS OF DIGITAL INTERFACES OF ARINC
CHARACTERISTIC SUBSYSTEMS (Continued)

ARINC Characteristic	Number	Inputs From	Speed	Digital		
				Number	Outputs To	Speed
709-4 DME	2	720 DFFSS		2	Freq. (035) 6 words Dist (201) sec Dist (202)	LO
710-3 ILS	2	720 DFFSS		1	AFCS	LO
				1	Data Utilization Devices, e.g. EFIS	LO
711-4 VOR	2	720 DFFSS		2		LO
712-3 ADF	2	720 DFFSS		2		LO
716-3 VHF	2	720 DFFSS			None	
717-3 DEFDARS	24			1		
718-3 ATCRBS/DABS	2	720 DFFSS		1	Crew Terminal #1	
	2	706 DADS		1	DAPS OUTPUT	
	2	704/702 IRS/FMC				
	1	Crew Terminal #2				
	1	Spare Data Input #2				
	1	Spare Data Input #3				
720-1	2	Spare		2		
	1	2nd 720 DFFSS	LO	1	2nd 720 DFFSS	
723	1	706 DADS		1	Caution Advisory Comp.	
	1	707 Radio Alt.				
	1	710 ILS Receiver				
	1	704/705				

TABLE 5-25. SYNOPSIS OF DIGITAL INTERFACES OF ARINC
CHARACTERISTIC SUBSYSTEMS (Continued)

ARINC Characteristic	Number	Inputs From	Speed	Digital		
				Number	Outputs To	Speed
725 Control Panel				2		LO
725 EFI Signal Generator	2	FAC	LO	1	Maintenance	
	2	726 FWC	HI			
	2	712 ADF	LO			
	2	725 Control Panel	LO			
	2	703 TCC	LO			
	2	702 FMC	HI			
	3	701 FCC	LO			
	3	704 IRS	HI			
	3	710 ILS	LO			
	2	709 DME	LO			
	2	711 VOR	LO			
	3	707 Radio Alt.	LO			
	2	708 WX Radar	453			
	2	706 DADS	LO			
	3	727 MLS	LO			
	2	Glare Shield Controller	LO			
726-1 Flight Warning Computer System	2	707 Radio Altimeters	LO			
	2	706 DADS	LO			
	3	704/705 IRS/AHRS	HI			
	2	710 ILS	LO			
	2	Engine/Fuel Airframe				
	1	2nd 726-1 FWCU		1	2nd 726-1 FWCU	
	1	Warning/Caution Display		2	Warning Caution Display	LO
	3	701 FCC		1	AIDS/Maintenance	LO
	1	703 TCC				
	2	Spare	LO			
729-1 Analog/Discrete Data Converter Set				2		HI
730-3 Airborne Separation Assurance System	2	Transponder Control		1	Crew Terminal #1	
	2	706 DADS	LO	1	453 (MP5D & MP5E)	
	1	707 Radio Altimeter				
	2	704/702 IRS/FMC				
	1	453 (MP5G & MP5H)				
	3	Spare				
599 Mark 2 OMEGA Navigation System	1	2nd 599 OMEGA		5		
	1	706 DADS				
	1	702 FMCS				
	1	704 IRS				

5.7 REFERENCES.

1. Lloyd, David K., and Lipow, Myron, Reliability: Management, Methods, and Mathematics, Second Edition, Published by the Authors, Redondo Beach, California, 1977.
2. Kaufmann, A., Grouchko, D., and Cruon, R., Mathematical Models for the Study of the Reliability of Systems, Academic Press, New York, 1977.
3. Myers, Glenford, J., Software Reliability: Principles and Practices, John Wiley & Sons, New York, 1976.
4. Gephart, L. S., et al, "Software Reliability: Determination and Prediction", AFFDL-TR-78-77, University of Dayton, June 1978.
5. Avizienis, Algirdas, "Fault-Tolerance: The Survival Attribute of Digital Systems", Proceedings of the IEEE, Vol. 66, No. 10, Pages 1109-1125, October 1978.
6. Hitt, Ellis F. and Bridgman, Michael S., "Formulation of a Methodology for Evaluation of the Mission Effectiveness of Fault Tolerant Integrated Control Systems on a Single Aircraft", Contract F33615-76-C-3145, Item No. 0002, Sequence No. AFFDL/AC, Request No. 50, Battelle Columbus Laboratories, December 1979.
7. Poyneer, Robert D., and Cunningham, Thomas B., "Fault Tolerant Digital Flight Control Using Analytic Redundancy", NAECON '77 Record, Pages 111-120.
8. Shapiro, E. Y., "Software Techniques for Sensor Redundancy Management of Flight Control Systems", Journal of Aircraft, Vol. 14, No. 7, Pages 632-683, July 1977.
9. Cunningham, T. B., et al, "Fault Tolerant Digital Flight Control With Analytical Redundancy", AFFDL-TR-77-25, Honeywell, Incorporated, May 1977.
10. Clark, R. N., "Instrument Fault Detection", IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-14, No. 3, Pages 456-465, May 1978.
11. Masreliez, C. Johan, and Bjurman, Bo E., "Fault Tolerant System Reliability Modeling/Analysis", Journal of Aircraft, Vol. 14, No. 8, Pages 803-808, August 1977.
12. Bosch, J. A., and Kuehl, W. J., "Reconfigurable Redundancy Management for Aircraft Flight Control", Journal of Aircraft, Vol. 14, No. 14, Pages 966-971, October 1977.
13. Boudreau, J. A., "Impact of CCV Requirements on Flight Control System Design", Journal of Aircraft, Vol. 14, No. 11, Pages 1051-1059, November 1977.
14. Dade, W. W., Edward, R. H., Katt, G. T., McClellan, K. L., and Shomber, H. A., "Flight Control Electronics Reliability/Maintenance Study", NASA CR-145271, Boeing Commercial Airplane Company, Seattle, Washington, December 1977.

15. Conn, Ralph B., et al, "Definition and Trade-Off Study of Reconfigurable Airborne Digital Computer System Organizations", NASA CR 132552, Ultrasystems, Incorporated, Newport Beach, California, November 1974.
16. Advanced Control Technology and its Potential for Future Transport Aircraft, NASA TMX-3409, Dryden Flight Research Center, Edwards, California, August 1976.
17. Hofmann, L. Gregor, and Clement, Warren F., "Vehicle Design Considerations for Active Control Applications to Subsonic Transport Aircraft", NASA CR 2408, Systems Technology, Incorporated, Hawthorne, California, August 1974.
18. Frazzini, Ronald, and Vaughn, Darrel, "Analysis and Preliminary Design of an Advanced Technology Transport Flight Control System", NASA CR-2490, Honeywell, Incorporated, Hopkins, Minnesota, March 1975.
19. Maybeck, Peter S., "Failure Detection Through Functional Redundancy", AFFDL-TR-74-3, January 1974.
20. Hartman, G. L., Harvey, C. A., Stein, G., et al, "Digital Adaptive F-8C Control Laws", Final Report NAS1-13358, July 1975.
21. Kerr, T. H., "A Two Ellipsoid Overlap Test for Real Time Failure Detection and Isolation by Confidence Regions", IEEE Conference on Decision and Control, November 1974, Phoenix, Arizona.
22. "Advanced Fighter Digital Flight Control Definition Study", MCAIR Purchase Order 240020, Honeywell, Incorporated, March 1975.
23. Mehra, R. K., and Peschon, J., "An Innovative Approach to Fault Detection and Diagnosis in Dynamics Systems", Automatics, Vol. 7, Pages 637-640, 1971.
24. Stein, G., and Hartman, G. L., "F-8C Adaptive Control Extensions", NASA Contract NAS1-13383, June 1976.
25. Wensley, J. H., et al, "Design Study of Software-Implemented Fault-Tolerance (SIFT) Computer", SRI International, Interim Technical Report 1, NAS1-13792, June 1978.
26. Smith, T. B., Hopkins, A. L., et al, "A Fault-Tolerant Multiprocessor Architecture for Aircraft", Vol. 1, The Charles Stark Draper Laboratory, Incorporated, NASA CR 3010, July 1976.
27. "Mark 33 Digital Information Transfer System (DITS)", ARINC Specification 429-6, Aeronautical Radio, Incorporated, February 10, 1982.

28. "Very High Speed Data Bus", Draft 2 of Project 453, Airlines Electronic Engineering Committee Letter #78-121/SAI-82, Aeronautical Radio, Incorporated, September 12, 1978.
29. "Aircraft Internal Time Division Command/Response Multiplex Data Bus", MIL-STD-1553B, Department of Defense, September 21, 1978.
30. "Notice 1, Aircraft Internal Time Division Command/Response Multiplex Data Bus", United States Air Force, February 12, 1980.
31. "IEEE Standard Digital Interface for Programmable Instrumentation", IEEE Std. 488-1978, The Institute of Electronic Engineers, Incorporated, November 30, 1978.
32. PDP-11 Peripherals Handbook, Digital Equipment Corporation, Maynard, Massachusetts, (1976), Pages 5.1 to 6.1.
33. "CPU Boards--A Matter of Bits, Buses, and BASIC", EDN, 22(21), 1977, Pages 144-150.
34. Warren, Carl, "Compare μ C-Bus Specs to Find the Bus You Need", EDN, June 10, 1981, Pages 141-153.
35. Hitt, Ellis F., and Broderson, Robert L., "Integrated Control Core Software Concept Study", AFWAL-TR-81-3141, Battelle Columbus Laboratories, September 1981.
36. Kurlak, Raymond P., and Chobot, J. Robert, "CPU Coverage Evaluation Using Automatic Fault Injection", AIAA Paper 81-2281, 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.
37. McGough, John G., Swern, Fred, and Bavuso, Salvatore J., "Methodology for Measurement of Fault Latency in a Digital Avionic Miniprocessor", AIAA Paper 81-2282, 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.
38. "Reliability Advancement for Electronic Engine Controllers", Vol. 1, AFWAL-TR-80-2063, Hamilton Standard Division of United Technologies Corporation, June 1980.
39. Erwood, R. G., McCorkle, R. D., Parente, J. J., et al, "Digital Flight Control Redundancy Management System Development Program", AFFDL-TR-79-3050, Vol. II, Boeing Aerospace Company, May 1979.
40. "Hall, Robert F., "Monitoring for Failure Conditions in Individual ("Single-String") and Redundant Computerized Systems or Subsystems", Appendix B, Fifth Draft RTCA DO-XXX, RTCA SC-145, February 20, 1981.
41. "MIL-STD-1553 Multiplex Applications Handbook", Air Force Systems Command, Aeronautical Systems Division, ENASD, Wright-Patterson AFB, Ohio, 45433, May 1980.

42. Spradlin, Richard E., "The 757/767 Flight Management System Laboratory Test Program", 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.
43. Chun, Randall K., "ARINC 429 Digital Data Communications on the Boeing 757 and 767 Commercial Airliners", AIAA Paper 81-2267, 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.
44. "Airplane System Design Analysis", Advisory Circular 25.1309-XX, Federal Aviation Administration.
45. Mulcare, D. B., Ness, W. G., McCarty, J. M., Richards, J. M., et al, "Industry Perspective on Simulation Methods and Research for Validation and Failure Effects Analysis of Advanced Digital Flight Control/Avionics", NASA CR 152234, Lockheed-Georgia Company, January 22, 1978.
46. "Mark 1 Air Transport Area Navigation System", ARINC Characteristic 581, Aeronautical Radio, Incorporated, June 30, 1970.
47. "Mark 13 Area Navigation System", ARINC Characteristic 583, Aeronautical Radio, Incorporated, September 23, 1976.
48. "Flight Control Computer System", ARINC Characteristic 701, Aeronautical Radio, Incorporated, March 1, 1979.
49. "Flight Management Computer System", ARINC Characteristic 702-1, Aeronautical Radio, Incorporated, January 29, 1980.
50. "Thrust Control Computer System", ARINC Characteristic 703, Aeronautical Radio, Incorporated, March 1, 1979.
51. "Inertial Reference System", ARINC Characteristic 704-3, Aeronautical Radio, Incorporated, August 19, 1981.
52. "Attitude and Heading Reference System", ARINC Characteristic 705-3, Aeronautical Radio, Incorporated, March 13, 1981.
53. "Mark 5 Subsonic Air Data System", ARINC Characteristic 706-2, Aeronautical Radio, Incorporated, February 27, 1981.
54. "Radio Altimeter", ARINC Characteristic 707-3, Aeronautical Radio, Incorporated, March 12, 1981.
55. "Airborne Weather Radar", ARINC Characteristic 708-2, Aeronautical Radio, Incorporated, February 20, 1981.
56. "Mark 5 Airborne Distance Measuring Equipment", ARINC Characteristic 709-4, Aeronautical Radio, Incorporated, April 3, 1981.
57. "Mark 2 Airborne ILS Receiver", ARINC Characteristic 710-3, Aeronautical Radio, Incorporated, August 18, 1980.

58. "Mark 2 Airborne VOR Receiver", ARINC Characteristic 711-4, Aeronautical Radio, Incorporated, April 10, 1981.
59. "Airborne ADF System", ARINC Characteristic 712-3, Aeronautical Radio, Incorporated, April 15, 1981.
60. "Mark 2 Omega Navigation System", ARINC Characteristic 599, Aeronautical Radio, Incorporated, September 25, 1981.
61. "Airborne VHF Communications Transceiver", ARINC Characteristic 716-3, Aeronautical Radio, Incorporated, September 25, 1981.
62. "Flight Data Acquisition and Recording System", ARINC Characteristic 717-3, Aeronautical Radio, Incorporated, March 27, 1981.
63. "Aircraft Integrated Data System (AIDS)--Mark 2", ARINC Characteristic 573-7, Aeronautical Radio, Incorporated, December 2, 1974.
64. "Mark 3 Air Traffic Control Transponder (ATCRBS/DABS)", ARINC Characteristic 718-3, Aeronautical Radio, Incorporated, September 10, 1981.
65. "Digital Frequency/Function Selection for Airborne Electronic Equipment", ARINC Specification 720-1, Aeronautical Radio, Incorporated, July 1, 1980.
66. "Ground Proximity Warning System", ARINC Characteristic 723-1, Aeronautical Radio, Incorporated, August 15, 1981.
67. "Electronic Flight Instruments (EFI)", ARINC Characteristic 725-1, Aeronautical Radio, Incorporated, September 5, 1980.
68. "Flight Warning Computer System", ARINC Characteristic 726-1, Aeronautical Radio, Incorporated, September 10, 1981.
69. "Airborne MLS Receiver, Part 1, Aircraft Installation Provisions", ARINC Characteristic 727, Aeronautical Radio, Incorporated, November 16, 1979.
70. "Analog and Discrete Data Converter System", ARINC Characteristic 729-1, Aeronautical Radio, Incorporated, September 10, 1981.
71. "Airborne Separation Assurance System", ARINC Characteristic 730-3, Aeronautical Radio, Incorporated, April 3, 1981.
72. "Electronic Chronometer", ARINC Characteristic 731, Aeronautical Radio, Incorporated, October 31, 1980.
73. Kuhlberg, J., and Zimmerman, W., "Flight Testing of an All Electronic Propulsion Control System", AIAA-80-1147, AIAA/SAE/ASME 16th Joint Propulsion Conference, 1980.

74. Dunn, Gregory L., and Leong, Patrick, "Conceptual Design of an Integrated Power and Avionics Information System", Boeing Military Airplane Company, "Proceedings of the IEEE 1981 National Aerospace and Electronics Conference, Vol. 3, Pages 954-962.
75. "Power System Control Study, Phase I, Integrated Control Techniques", AFAPL-TR-79-2084, Vought Corporation, October 1979.

SECTION SIX
CREW WORKLOAD EVALUATION

TABLE OF CONTENTS

	Page
SECTION 6	6-1
6. CREW WORKLOAD EVALUATION	6-1
6.1 Human Factors Engineering Goals	6-1
6.1.1 Safety and Crew Acceptability	6-1
6.1.2 Current Revolutionary Changes in Crew Interface	6-2
6.1.3 Greater Future Design Commonality	6-3
6.2 Regulatory Requirements	6-3
6.2.1 Minimum Crew Determination	6-3
6.2.2 Overall Flight-Deck Evaluation	6-5
6.3 Current Workload Measurement Techniques	6-7
6.3.1 Subjective Measures	6-7
6.3.2 Performance Measures	6-7
6.3.3 Physiological and Biological Measures	6-8
6.4 Automation and System Effectiveness	6-9
6.4.1 Impact on the Pilot	6-10
6.4.2 System Effectiveness	6-12
6.4.3 Changed Pilot Roles	6-14
6.4.4 Human Needs and Satisfaction	6-16
6.5 Future Workload Measurement Problems	6-17
6.5.1 Evaluation of Stress	6-17
6.5.2 Sources of Pilot Error	6-19
6.5.3 Definitions of Good Design	6-20
6.5.4 Relation of Workload to Safety	6-21
6.6 Recent Certification Programs Overview	6-23
6.6.1 Systems Analysis	6-23
6.6.2 Synthetic Performance Measures	6-25
6.6.3 Flight Test	6-29
6.6.4 Remaining Problems	6-35
6.6.5 Need to Expand the Variety of Procedure	6-35
6.6.6 Classification of Workload Assessment Methods	6-38

TABLE OF CONTENTS (Continued)

	Page
6.7 Recommended Applications	6-42
6.7.1 Specific Procedures	6-42
6.7.2 Combined Methods	6-45
6.8 References	6-47
APPENDIX	

LIST OF ILLUSTRATIONS

Figure		Page
6-1	Representative Elements in Goal-Effective System Operation	6-13
6-2	The Pilot-Aircraft-Environment Information Flow	6-18
6-3	Relation of High and Low Pilot Workload and Safety	6-22
6-4	DC-9-80 Comparison of Equipment Interface Workload with DC-9-50 When Autothrottles are Inoperative	6-28
6-5	Varieties of Digit Entry Devices	6-36
6-6	Inventory of Available Workload Methods and Issue/Application Situations	6-43
6-7	Population of Methods of Evaluating Pilot Workload	6-44

SECTION 6

6. CREW WORKLOAD EVALUATION

6.1 HUMAN FACTORS ENGINEERING GOALS.

Human Factors Engineering aims to integrate effectively the human components into various elements and operations of the aviation system. Specifically, in the case of civil airline aircraft design, this means that hardware, procedures, and training must be developed so that the possibility of accidents arising from human error or degraded performance is minimum (a safety). Also, it means that pilots, maintenance personnel, and other human elements such as flight engineers, managers, cabin attendants, cargo handlers, and the like are provided suitable work environments (acceptability).

6.1.1 Safety and Crew Acceptability.

The essence of human factors engineering is that human strengths and limitations should dictate the design of man-machine interfaces and the design of the operator/operational procedure of advanced systems. How well such considerations have been incorporated in the design of aviation systems and, hence, how safe and acceptable those systems are for human operators is an empirical matter (i.e., one that must be tested). The large literature of human factor studies provides handbook answers on best design of many subparts of systems (e.g., particularly principles of good display design and conventions of knob and control coding). When higher levels of automation are introduced, handbook design solutions are less applicable, and any complex man-machine systems, such as a new flight deck design, may extend beyond our ability to evaluate based on past studies. To determine that the new design is both safe (supportive of the required human performances and resistant to error) and acceptable (not unduly fatiguing or demanding of concentration, unusual skill or strength, etc.), the new design must be evaluated in use.

In practice, new flight deck designs are evaluated in two general forms of crew workload studies. Before the aircraft is ready to be flown, portions of the design/procedure package are tested in similar aircraft and in ground-based procedural simulator mock-ups and training devices. Once the aircraft is ready, actual flight tests are conducted before any new equipment or procedures are placed in commercial service. This chapter summarizes these test methods and leads to recommendations for application of current and new workload test methods. Hence, both evaluation procedures previously applied successfully and newer measures that have become available and are now ready for first commercial aircraft applications are covered.

From a system viewpoint, workload is the work required, the sum of all the human tasks that must be accomplished to realize the goals of the system (including the pilot mental performances, the events that must occur within the crew members, if those human tasks are to be completed appropriately). Systems level analyses can reveal many of the task demands imposed by a particular flight deck design, but not necessarily all of them. Because of the degree of complexity present in the design and implementation of cockpit technology, assurance that all workload demands are known comes only with final test flying. Similarly, standard operating procedures developed for pilots for a new design may clarify nearly all the information that the crew members must search out and accept, nearly all the thought processes that may be required to process that input data to appropriate decisions, and nearly all

the motor outputs necessary to operate the aircraft safely. However, empirical test is necessary to assess the magnitude and acceptability of the performances required of the pilot — how difficult or stressful, how monotonous or challenging, how time-consuming or quick — the various activities may be for the pilot. These tests are the subject of crew workload evaluations.

Individual differences in pilots make evaluation of workload more difficult. Among the individual factors influencing how difficult work is or how acceptable task demands may be are: skill level, recency of practice, motivation to perform, health status and fatigue, transitory emotional state, and such group psychological factors as inter-pilot confidence, ease of communication, and willingness to assist. Because of these factors that vary among pilots in the population and vary from time to time with the same pilot, workload tests must be repeated and must sample a variety of crews as well as test situations to provide a proper data base against which decisions/changes can be made and evaluated.

Beyond the specific features of the flight deck design, the prospective operating environment of the new aircraft has a major influence on workload. Aircraft aerodynamics (including handling qualities) and the weather environment may determine handling ease and the degree of reliance on automation and outside help from Air Traffic Control (ATC) that is necessitated in order to complete the mission. The nature of the ATC system itself and the availability of navigation aids and precision approach guidance will affect workload. Noise abatement requirements can complicate flight routes and force pilots to use reduced power settings. Scheduling considerations and the changing economic environment can add stress and require pilots to operate with less freedom of choice, so as to reduce fuel consumption or attain on-time performance. Because of these and related considerations, pilot workload evaluations must take into account more than the changed flight deck features and the capacities of the prospective crew members. Workload is the final result of equipment, procedures, human variables, and total environment. Its assessment, then, is a major activity, one that requires careful planning and use of the best available technology.

6.1.2 Current Revolutionary Changes in Crew Interface.

The 1980's and 1990's generation of flight deck digital systems represents one of the rare "revolutionary" changes in cockpit design. Twenty-five years ago, the early turbojet transport designs changed the pilot tasks and potentials for error due to both the changed aerodynamic properties of the aircraft (dutch roll, yaw tendencies, sink rate recovery techniques, etc.), and the greatly changed propulsion controls and indicator systems. At that time, each aircraft was designed with different cockpit systems, and a high degree of commonality was not present.

Subsequent transports showed "evolutionary" changes to the present, with the exception that several highly successful designs eliminated the third crew position with sidefacing console of systems displays and controls. That change occasioned major requirements for workload and safety testing. Otherwise, most design changes were incremental and did not produce a requirement for extensive test of pilot performance issues.

The present introduction, all at one time, of new aircraft with both changed flying/handling qualities and radically changed flight deck designs is the next successive "revolution." The B-767, 757 designs include a changed Electronic Flight Instrument System (EFIS), an Integrated Engine-Indicating and Crew Alerting System (EICAS), and a computer-based Flight Management System with navigation and

autopilot capabilities (FMS). This change is "revolutionary" and equivalent in potential impact on pilot performance and error potential to the initial turbojet design changes or the shift from three crew to two in the DC-9 and B-737 concepts.

6.1.3 Greater Future Design Commonality.

The new B-767, 757 computer-based designs constitute a major step toward common flight deck designs for different aircraft types. The individual manufacturer has an obvious incentive to promote commonality, and customer airlines favor commonality due to training and maintenance (spare parts) simplification. Beyond those concerns, the government has directly promoted flight deck design commonality in the present generation of new aircraft. First, the Federal Aviation Administration (FAA) supported industry-wide development of a common set of caution and warning requirements leading to the new EICAS. Also, federal development studies with the NASA Langley Terminal Configured Vehicle (TCV), a modified B-737, gave pilot performance data to all manufacturers leading to the new EFIS design. In addition, Department of Defense programs, such as the Air Force Digital Avionics Integrated System (DAIS) served to give avionics/ instrument manufacturers common interface and performance standards.

For the reasons cited above, the probability is high that the approved designs for EFIS, EICAS, and FMS in the B-767, 757 programs will have the effect of setting new baseline standards for other transport aircraft. It is not implied that workload measurement techniques and pilot performance evaluations carried out in earlier new technology aircraft test programs, (e.g., L-1011, B-747) were in any way inadequate. Rather, the importance of the present test programs is enhanced because of both the magnitude of the changes in pilot equipment and procedures and the likely pattern-setting influence of the results. This enhanced importance requires that the most careful consideration be given to the issues raised by the innovations and changed pilot performance requirements and that test methods for evaluation of these subsequent parallel designs be selected in the light of full knowledge of the rapidly developing technology of workload measurement.

The logic of the present situation is that flight deck equipment and task demands placed on the pilot are being so changed that the knowledge gained from past service experience is inadequate to allow theoretical predictions of pilot performance and error potential. At the same time that varied new issues are raised by design changes, the methods of evaluating pilot workload stress and human error potential are expanding in variety and power to assist in the needed evaluations.

More testing is needed because of the new systems being installed in a commercial transport cockpit for the first time and because of concerns relating to integration of the pilot with these units, not because of crew size considerations. The fact that advances are being made in pilot performance measurement techniques at the same time that these advances are most needed gives the opportunity to try out workload procedures that, up to now, have been used only in military and experimental programs.

6.2 REGULATORY REQUIREMENTS.

6.2.1 Minimum Flight Crew Determination.

Assessment of crew member workload as an element of new aircraft certification is provided for in Federal Aviation Regulations (FAR) Part 25, "Airworthiness

Standards: Transport Category Airplanes," (reference 1). The controlling statements found in FAR 25.1523, "Minimum Flight Crew," and Appendix D of FAR Part 25, with an effective date of February 1, 1965, represent a recodification of the earlier Civil Aeronautics Regulation (CAR) 4b which had stated:

"Minimum Flight Crew - The minimum flight crew shall be established by the Administrator as that number of persons which he finds necessary for safety in the operations authorized under Section 4b.721. This finding shall be based upon the workload imposed upon individual crew members with due consideration given to the accessibility and the ease of operation of all necessary controls by the appropriate crew members."

Restated in the new FAR Part 25, the requirements are as follows:

"The minimum flight crew must be established so that it is sufficient for safe operation, considering:

- (a) The workload on individual crew members;
- (b) The accessibility and ease of operation of necessary controls by appropriate crew member; and,
- (c) The kind of operation authorized under 25.1525.

The criteria used in making the determinations required by this section are set forth in appendix D."

FAR Part 25, appendix D, list six functions and ten factors to be considered in determining the minimum flight crew. The criteria are as follows:

Part 25, Airworthiness Standards: Transport Category Airplanes

APPENDIX D

Criteria for determining minimum flight crew. The following are considered by the agency in determining the minimum flight crew under 25.1523:

- a. Basic workload functions. The following basic workload functions are considered:
 - (1) Flight path control.
 - (2) Collision avoidance.
 - (3) Navigation.
 - (4) Communications.
 - (5) Operation and monitoring of aircraft engines and systems.
 - (6) Command decisions.
- b. Workload factors. The following workload factors are considered significant when analyzing and demonstrating workload for minimum flight crew determination:
 - (1) The accessibility, ease, and simplicity of operation of all necessary flight, power, and equipment controls, including emergency fuel shutoff valves, electrical controls, electronic controls, pressurization system controls, and engine controls.

- (2) The accessibility and conspicuity of all necessary instruments and failure warning devices such as fire warning, electrical system malfunction, and other failure or caution indicators. The extent to which such instruments or devices direct the proper corrective action is also considered.
- (3) The number, urgency, and complexity of operating procedures with particular consideration given to the specific fuel management schedule imposed by center of gravity, structural or other considerations of an airworthiness nature, and to the ability of each engine to operate at all times from a single tank or source which is automatically replenished if fuel is also stored in other tanks.
- (4) The degree and duration of concentrated mental and physical effort involved in normal operation and in diagnosing and coping with malfunctions and emergencies.
- (5) The extent of required monitoring of fuel, hydraulic pressurization, electrical, electronic, deicing, and other systems while enroute.
- (6) The actions requiring a crew member to be unavailable at his assigned duty station, including: Observation of systems, emergency operation of any control, and emergencies in any compartment.
- (7) The degree of automation provided in the aircraft systems to afford (after failures or malfunctions) automatic crossover or isolation of difficulties to minimize the need for flight crew to guard against loss of hydraulic or electric power to flight controls or other essential systems.
- (8) The communications and navigation workload.
- (9) The possibility of increased workload associated with any emergency that may lead to other emergencies.
- (10) Incapacitation of a flight crew member whenever the applicable operating rule requires a minimum flight crew of at least two pilots.

c. Kind of operation authorized. The determination of the kind of operation authorized requires consideration of the operating rules under which the airplane will be operated. Unless the applicant desires approval for a more limited kind of operation, it is assumed that each airplane certificated under this part will operate under IFR conditions.

6.2.2 Overall Flight-Deck Evaluation.

The minimum flight crew regulation, FAR 25.1523, provides the nexus for flight deck design evaluation as to conformance with good human factors design practice. In addition to FAR 25.1523, other provisions of Part 25 state various flight deck performance requirements with particular sections having to do with instruments,

displays, controls, and miscellaneous equipment such as seat restraints, electric protective devices, radios, fire extinguishers, etc. In the overall type certification process, however, it is in the application of the minimum flight crew section, 25.1523, that the most general check is required on the adequacy of human factors design.

The actual wording of the regulations does not specify any single or unitary definition of crew workload or state how workload is to be evaluated either quantitatively or qualitatively. Broad discretionary authority rests, then, with the FAA Administrator and his delegated Type Certification Board (TCB) to determine both the particular kind and amount of testing that is required to show regulatory compliance.

The enumeration of workload factors in Appendix D does, however, constitute a human factors checklist for good engineering design. Clearly pointing to the need for a comprehensive flight deck system evaluation are paragraphs calling for analysis and demonstrations of "the accessibility, ease, and simplicity of operation of all necessary flight, power, and equipment controls, . . ."; "the accessibility and conspicuity of all necessary instruments and failure warning devices . . ."; "the number, urgency and complexity of operating procedures . . ."; "the degree and duration of concentrated mental and physical effort . . ."; "the extent of required monitoring . . ."; "the actions requiring a crew member to be unavailable at his assigned duty station . . ."; "the degree of automation provided in the aircraft systems to afford (after failures or malfunctions) automatic crossover or isolation of difficulties . . ."; and so forth.

Since workload is not further defined in the regulation, the most acceptable working definition is that crew workload is the total task that each crew member must perform while flying the airplane. In the extensive literature of workload assessment, there is no single definition that has been found suitable for all possible uses in different design and assessment tests, and no single standard means of measuring workload by the aircraft industry exists today (reference 2).

As described in the FAA handbook, "Type Certification," the purpose of examination of flight deck arrangement and control operation is ". . . to insure that the flight crew will be able to perform all of its duties without unreasonable concentration, fatigue, and without the likelihood of incorrect operation " (reference 3). Simply stated, the design must support pilot performance and avoid design induced error potential. Since there are not extant human factors models of pilot performance from which positive behavior probabilities can be derived as a function of each design feature, many flight deck evaluations must be made by conduct of pilot performance tests and report of expert opinion. The need for this is noted in Paragraph 167 of the FAA handbook as follows:

- a. Purpose. A multiple-expert-opinion evaluation may be used for determining compliance of controversial qualitative flight test certification design and operational features. This type of evaluation should be employed on an agency-wide basis whenever regional personnel feel that the issue is subject to controversy, precedent-setting, and/or transcends the local region's concern, for which specific guidelines and criteria have yet to be developed and issued.

- (1) Determination of FAR compliance on an "equivalent level of safety basis" usually involves a qualitative analysis of an aircraft which possesses design features which do not meet the "letter of the regulation" or are not clearly covered in the applicable regulations.

AD-A133 222

VALIDATION OF DIGITAL SYSTEMS IN AVIONICS AND FLIGHT
CONTROL APPLICATIONS. (U) BATTELLE COLUMBUS LABS OH
E F HITT ET AL JUL 83 DOT/FAR/CT-82/115

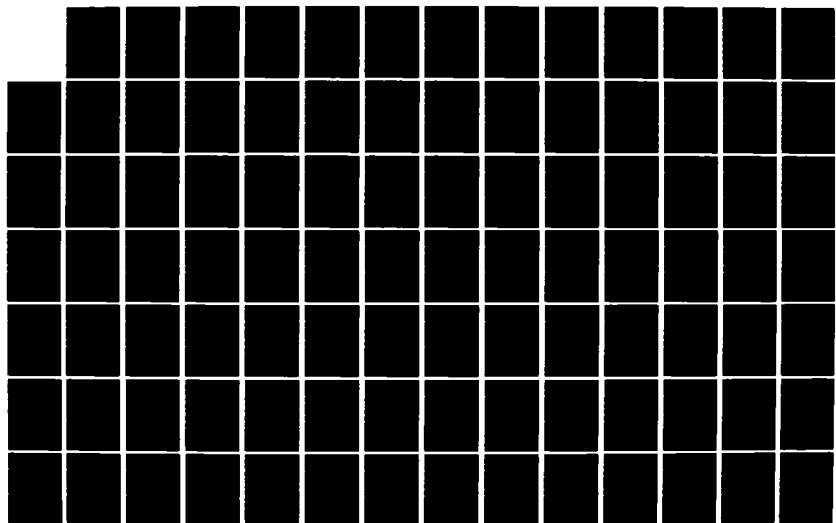
3/8

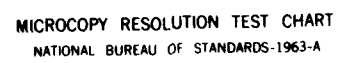
UNCLASSIFIED

DTFA03-81-C-00059

F/G 1/3

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

The FAA handbook, "Type Certification," also requires approval of aircraft flight manuals (paragraph 174) and development of a minimum equipment list (MEL) (paragraph 168). The manual must include operating limitations, and summary of all normal and emergency procedures, while the MEL establishes the equipment which is essential for safe operation. Clearly, flight test determinations on the adequacy of both flight manuals and MEL's require full examination of the flight deck design and crew procedures to determine the adequacy of design. Quantitative workload level is recognized as important, but also the rules focus on the particular operating problems and difficulties imposed by emergencies and equipment outages. Thus, the test program must be adequate to produce data on the design capacity to support pilot performance and avoid design induced errors, not merely the acceptability of the magnitude of task demands for the planned crew size.

6.3. CURRENT WORKLOAD MEASUREMENT TECHNIQUES.

Possible Measures Including Those Most Used in Past Certification Programs. Authorities agree that energy consumption measures show the pilot to be in light physical work (reference 4). Hence, the emphasis in past workload tests has been placed on mental workload. Most possible measures of such mental workload have, in fact, been applied to aircraft pilots, although none has proved so satisfactory as to gain universal acceptance. For such acceptance, it would be required that a measure of mental workload permit comparison to standards such that the task complex might be judged: (a) Overly demanding; (b) insufficiently stimulating; or (c) wrong in kind such that inattention or performance decrements are likely.

A general consensus of international experts has been summarized by saying that mental workload is a conceptual label for a complex of somewhat separate and independent pilot behaviors, such that it would be unreasonable to expect fully adequate assessment by any single measure (reference 5).

6.3.1 Subjective Measures.

In the case of crew complement determination and compliance with FAR 25.1523, the final decision in all recent certification programs has been reserved until the aircraft has been flown and evaluated by a panel of experienced pilots. While many other workload comparisons and tests may have preceded actual flight test, it appears that more assurance is derived from actual flight test than from simulator tests or time-and-motion type activity analyses. One reason for this assurance is that the experienced pilot after flying the new airplane can integrate all of his observational data and make a unitary subjective judgment that the new aircraft does or does not compare favorably in workload and acceptability for airline operation to other service proven aircraft. Subjective measures standardized on a questionnaire format or a Cooper-Harper type rating scale have this marked advantage of providing an overall integrated pass-fail assessment (reference 6). The disadvantage inherent in the method is, of course, the limitation that a pilot can only report on his experience and then only as recalled, not on the multiplicity of workload experiences that might have occurred under an infinite number of other circumstances.

6.3.2 Performance Measures.

Pilot performance is measured in both simulated and actual flight, and aircraft performance resulting from pilot activity is also evaluated (reference 7). The

most widely used performance measures in crew workload evaluation are comparisons between activities in the newly designed flight deck environment and activities in a service proven aircraft.

The general method requires construction of a highly detailed flight scenario covering what are believed to be workload critical phases of flight, such as final approach in instrument meteorological conditions. Such a scenario is based on actual time-and-motion analysis of pilots performing the particular flight phases in existing aircraft. Altitude, speed, and heading profiles are recorded in actual instrument approaches made in the comparison aircraft, thereby providing a detailed average record of total aircraft performance. Then, the molecular task performances of each crew member are tabulated and lined against the aircraft performance record. Each time a hand moves a control, each time the view is redirected, each ATC communication, etc., is timed for duration and placed in a time sequence as the operation proceeds. From these records, it is possible to integrate workload in the body activity channel, external vision availability, equipment interfaces workload, and communications.

Following construction of a flight scenario, comparison measures are obtained for the new design. This may be done in a real-time simulation with actual pilots moving simulator controls and going through simulated ATC procedures, or it may be a computer simulation. Using a complex model of the observed pilot actions and known dimensions and features of the existing aircraft flight deck, a computer model of the new design incorporating its changed physical arrangements and revised flight procedures may be constructed. Fast-time iteration of various flight profiles, weather, and ATC demands can produce measures representing similar body channel utilization and other workload demands for both the conventional and novel designs.

In past certification programs, comparison measures of this kind, derived from laborious activity analysis and computer iteration, have supplemented the subjective judgment of test pilots. In the case of the DC-9-80, for example, the computer data verified expected workload advantages of several added flight deck automation systems. These data were submitted to the TCB and served to support the approval judgments of the pilot panel that actually flew the new aircraft. Other performance measurement techniques include measurement in a simulator of the data input requirements of a particular design, measurement of the time required to identify and solve simulated failures or emergencies, and external vision studies made to determine the amount of outside visual field available. The term part-task simulation is applied to studies that do not involve both the control workload usual in flying and the perceptual, information processing, decision making, and crew coordination activities, but rather limit the test situation to one or another subdivision of the total of crew tasks. Alternatively, whole task simulation is the term reserved for performance measurement in an animated and largely operable simulator approaching the so-called "training level" simulator in capacity to duplicate the actual aircraft.

6.3.3 Physiological and Biological Measures.

From time to time, such physiological measures as heart rate, galvanic skin conductance, and brain electrical wave patterns have been proposed (reference 8). Despite continued research over the entire period of the 1960's and 1970's, no instances are known in which pilot physiological data were used in workload assessments by civilian type certification authorities. As indicated in preceding sections, principal reliance has been placed on the expert opinion and judgment of

pilots trained to fly the new aircraft, and also highly qualified in current airline aircraft. Secondary reliance has been placed on the activity analysis type of performance measures which yield numerical counts showing how busy, mentally and physically, each crew member was at each stage of high workload flight.

Mental workload, which cannot be directly observed in performance measures, has always been recognized as highly significant in aircraft pilotage. But as long as aircraft were flown by hand and information displays were segregated by place, indirect insight into what the pilot was perceiving and acting on could be obtained by observing his hand and eye movements. With increasing levels of control automation, and the more highly integrated flight information and caution and warning system (EICAS), this becomes less feasible. At the ultimate extreme, the pilot might fly the aircraft with observable motions merely following the autopilot/ autothrottle initiatives and watching centralized, forward displays. In such a case, performance measures as typically recorded provide less clear indications of how busy and how intensely stressed the pilot is (reference 9).

Many proposals have been advanced in recent years to obtain physiological data that would substitute for the role of activity analysis data in providing objective and numerical back-up to the subjective ratings given by pilots (references 10 and 11).

The military services here and abroad have been in the forefront of development in the physiological area, and have brought this experimental workload assessment technology out of the laboratory and into actual flight programs (reference 12). In addition, the most recent overseas transport cockpit development program, conducted by AIRBUS Industries and the French aviation authorities, has utilized recording of pilot physiological data to indicate level of stress/workload (reference 13). In view of all of these changes and developments, it appears that more emphasis will be given to this area in the future.

The problem of just how to use physiological measures is not fully solved (reference 14). But in view of the fact that there is no other objective way to record events within the pilot exposed to varying work demands and experiencing varying levels of job stress, it is widely believed that civil applications must be made of the diverse military and research techniques.

6.4 AUTOMATION AND SYSTEM EFFECTIVENESS.

The importance of developmental and certification testing of controls and displays provided to allow the air crew to interface with computer based flight deck systems has been recognized explicitly by both customer airlines and aircraft manufacturers (reference 15). In the case of airline avionics and control systems certificated before 1980, such testing has not presented an equivalent level of criticality because of the existence of an "established" system configuration. The exception to this point has come up in demonstrations of the effectiveness of two-pilot designs. For that exception, careful examination of certification programs has led to the conclusion that past test methods were valid and sufficient and that crew-complement decisions resulting from workload evaluations have been correct (reference 16).

Prior to the present decade, the conventional cockpit was based on regulatory requirements, widely accepted standardization practices sponsored by organizations such as the Society of Automotive Engineers (SAE), Aeronautical Recommended Practices, and industry conventions representing proven solutions to demands of the

operating environment. Gradually, computer elements were added in incremental fashion (e.g., flight director instrumentation), sensors and data acquisition were improved (e.g., radar altimeter, Distance Measuring Equipment (DME), INS-LORAN), and displays were processed electrically to produce more useful signals (e.g., all-counter altimeter). In the case of such improvements, favorable impacts on crew workload could be shown fairly readily since the quality of information presented to the pilot and its integration into convenient patterns were improved. Generally, there was no indication of concomittant workload penalties, particularly since raw-data instrumentation was retained for back-up and cross-check purposes, and the various improvements were introduced over a time period sufficient to allow for gradual pilot familiarization and acceptance.

There has been a speed-up of the pace of automation in the 1970's. In the L-1011 program (circa 1970) and some subsequent transport aircraft with advanced flight deck systems (e.g., the DC-9-80 of 1981), additional automation features were provided. In later L-1011's, the first standard installation was made of a flight-management system intended to provide for flight path optimization. Additionally, development work led to the completion of active control systems capable of improving the structural efficiency of newer aircraft.

6.4.1 Impact on the Pilot.

Crew workload impacts of these later developments are different in kind from those flowing from most earlier flight deck changes. For example, the usual operating mode of the pilot may become that of a systems monitor/decisionmaker as opposed to that of an active manual controller. Instead of an occasional coupled autopilot approach to low altitude, the pilot might experience routine takeoff to touchdown with hands off the controls and power levers. Furthermore, the computational algorithms necessary for flightpath optimization and active control efficiency are too complex to permit the pilot to conduct real-time confirmatory mental calculations. Hence, he cannot cross-check the correctness of the computed command information by referring to analog instrumentation with sufficient lead time to adjust computer generated actions as a "pilot in the loop."

Obviously, automation improvements such as those cited would reduce manual control workload while the computer features were in use. But what would be the workload when routinely employed automation was not selected for whatever reason — actual failure, suspected variable errors, or pilot option for manual practice. Beyond the question of manual control workload in a manual reversion mode of operation, it is necessary to ask: What are the mental workload impacts of the changed pilot role as systems monitor and mode selector?

Differences may be noted in the measurement techniques appropriate for evaluating observable versus covert crew workload. For example, compare the adoption of full-time autothrottles to the adoption of newer computer based thrust management systems. The autothrottles were intended to serve, during the high workload phase of final approach to landing, to unburden the pilot since manual power adjustments would not be required with each flap and gear reconfiguration. Verification of this workload reduction could be obtained by observing the frequency of hand adjustments and eye references to airspeed and rate of descent instruments. When such counts were made in the DC-9-80, it was demonstrated experimentally that the expected workload reduction was, in fact, obtained (reference 17).

In contrast to the problem of evaluating the workload impact of adding auto-throttles, much of the expected change in information processing demand on the pilot resulting from installation of digital avionics coupled to primary controls of the airplane is covert and difficult to observe for purposes of counting and timing in classic task analysis or time-and-motion study techniques. Hence, the workload evaluation procedures used successfully to confirm planned flight deck task demand reductions in earlier evaluation programs have reduced validity for newer systems.

The 1980's/1990's generation of new flight deck features, as typified by the B-767 flight management systems and ARINC series 700 avionics, has several additional potential workload impacts. Primary flight instrument displays are being changed from conventional electromechanical devices to flat panel displays presented on shadow-mask CRT's, liquid crystal displays, and plasma panels employing color coding. While under normal conditions, these displays are expected to provide large, high resolution, high-contrast images appearing to be visually superior and with information content greater than earlier instruments, it is not certain that critical tests involving worst case conditions can be conducted in exactly the same manner as they were before. Similarly, the new caution and advisory computer, (EICAS), replaces multiple, hard-wired separate alerts with an annunciator system providing selective display, voice advisory capacity, prioritization/inhibit, automatic clearing (declutter modes), and use of color alphanumeric and graphic presentations (reference 18). While more information, particularly guidance for corrective actions, is an inherent advantage of such a digital-based system, choice of a best method of testing to verify use in high workload conditions must be considered in light of possible accompanying pitfalls, such as color vision defects of the pilot population and the possibility of incorrect prioritization in unpredicted but possible multiple failure situations.

Fortunately, design guidelines are available for many of the questions that may be related to operability of the newer displays in the highly varied ambient conditions found on flight decks. When such proposed standards are available, as in the case of cockpit color displays (reference 19), the evaluation questions are simplified.

Finally, there is a third general area of concern to the planning of workload evaluations with newer digital flight-deck systems. This is in addition to the methodological problems present with the shift of major pilot activity from overt, observable activities to more emphasis on covert monitoring and mental work, and the additional problems related to the employment of computer generated displays. While it may be determined that special tests are needed to verify utility of the Cathode-Ray Tube (CRT) displays in such conditions as unusual vibration, turbulence, light adaptation extremes, lightning, etc., the most difficult of all certification test requirements of the pilot interfaces may be those related to demonstrating design integrity. In a manual system, the operator confronted with untoward events relies on past experience and training and direct observation of environmental information and instruments. At higher levels of automation, there is a less direct information loop with the result that unplanned events not included in the operational scenarios covered by available software (logic trees and fault modes) create special problems. To a degree, blind reliance on the system is required since the capacity manually to compute and control is minimal.

6.4.2 System Effectiveness.

A comprehensive tabulation of system, personnel, and personal elements that determine the effectiveness of man-machine systems such as aircraft flight decks is presented by figure 6-1. The recommended approach to evaluate any given system is to begin with clear and explicit definitions of the system goals and the scenarios, task sequences, and actions necessary to accomplish those goals. The system can then be analyzed to look for man-task compatibility, conflicts, and potential problems. According to Parks (reference 14):

"Two generic types of analysis have been found to be beneficial in meeting applied goals. One is system oriented, to define and maintain a functional framework for system goals, requirements and operations, and for defining the human operator role in the operating system. The second is more detailed, concerned with assuring that the evolving requirements for man-machine interface fit the capabilities of the operator and observe human limitations and constraints. The latter fits the more traditional man-machine concepts regarding:

- Sensing
- Information processing
- Decision making
- Control actions
- Environmental and situational variables, e.g.,
 - work environment
 - work area layout
 - task complexity and compatibility
 - work-rest cycles
 - short term - long term variables and effects
 - motivational influence
 - off duty activities and influences"

Further, the tasks assigned the pilot may be classified for ease in planning evaluations. The most basic division is between tasks that are primarily physical and other tasks primarily nonphysical. It should be recognized, however, that physical tasks have mental correlates and vice versa, so that there are no purely physical or purely mental work elements. A rough ranking of the proportions blended in different work tasks may be proposed as follows:

1. Producing force (muscular work)
2. Sensory - motor coordination (tracking)
3. Seeking information (scanning, feeling)
4. Converting information to action (decision)
5. Creating information (thinking, solving)

The four levels of primarily nonphysical work are often described as sensory-perceptual, cognitive-information processing, decisionmaking between alternatives, and communication-action output. Each of these types of mental work has impacts on internal body systems and may be reflected in glandular secretion, changes in respiration or circulation, and cumulative fatigue. Because of his complexity

FACTORS IN SYSTEM EFFECTIVENESS

SYSTEM	PERSONNNEL	PERSONAL
OPERATONS	SELECTION	CRAFTMANSHIP/ PROFESSIONALISM
-FUNCTIONS	TRAINING	
-TASKS		
-PROCEDURES	STRESS/STRAIN	ACCEPTABLE WORK OF PERFORMANCE
-SEQUENCING		
-TIME	SKILL	
-ACCURACY		DEGREE OF INVOLVEMENT/ MOTIVATION
-RELIABILITY	FATIGUE	
-CRITICALITY		
-CRITERION CHANGES	WORK-REST CYCLE	JOB STAISFACTION
-SAFETY		
-OTHER	SHIFT	TRUST
		- IN EQUIPMENT
FEATURES	OTHER	- IN PERSONNEL
-SIGNALS		SHIFT
-RESOLUTION		
-DYNAMICS		AGE
-COMPLEXITY		
-MAN-MACHINE INTERFACE		LONG TERM
REQUIREMENTS		TOLERANCE/EFFECTS
LOCATION/ACCESS		
ARRANGEMENT		
OTHER		LIFE STYLE
-REQUIRED ACTIONS		-RECREATION
-REQUIRED INFORMATION		-COMMUTING
FOR ACTION		-SLEEP
-ERROR IMPACT		-OTHER
-MAN-MACHINE		
TASK ALLOCATION		GROUP-SOCIAL
-ENVIRONMENT		INFLUENCES
-OTHER		
OTHER		OTHER

FIGURE 6-1. REPRESENTATIVE ELEMENTS IN GOAL-EFFCTIVE SYSTEM OPERATION

of tasks and task impacts on the pilot, it is desirable to achieve a recommended program of workload evaluation methods tailored to appraise particular task elements that are most prominent in particular systems of interest. In the following paragraphs, these considerations will be directed toward the situation of the pilot using the new computer based systems.

6.4.3 Changed Pilot Roles.

The addition of computer based digital avionics, flight management systems, EICAS, and primary electronic flight instrument displays has major implications for the workload of the flight crew members. In the early days of long-range commercial flight, the pilot functioned primarily in a "tracking" mode (reference 20). For example, the clipper boat operations studied for workload often involved flight at what would now be considered extremely low altitudes and of long duration with no autopilot. Brief inattention or release of pressure on the controls meant an accident.

At the current, time there is an important trend to change the pilot role from tracking to monitoring. In contrast to earlier flight demands, the advanced aircraft cockpit, with a full complement of integrated computer driven flight deck systems, presents a radically changed work task for the pilot. In place of continuous manual tracking, during the cruise (en route) phase of flight, the major tasks are display and systems monitoring and supervision. It is a well-proven principle of experimental psychology that man can become very efficient at continuous tracking. The primary requirements are an interesting task (incentive) and good feedback (clear information for corrective action). Tracking performances are often poor in the novice, but as skill develops, and the phenomenon of "over-learning" proceeds, accurate tracking becomes easier and semiautomatic, so that man can track and think of other matters simultaneously.

Monitoring is quite different from manual tracking. Generally, man is a poor monitor (reference 21). Particularly when scanning displays on which little is happening and there is nothing to do, monitoring performance degrades. Often this is described as the difficulty of maintaining vigilance when signals are very infrequent. After a long period of searching for information that does not come, a valid signal may be missed even though in magnitude it far exceeds the just noticeable difference or perceptual threshold.

Monitoring is one of the many situations where minimum workload is not necessarily the best workload for human performance assurance. The extensive literature of studies on vigilance shows that there is an optimum signal frequency for many signal detection tasks and that performance may become more reliable if the task has "designed-in" events and response requirements.

Supervision is another major pilot role, and one that may be conducted with higher or lower reliability depending upon circumstances. "Supervision" implies that final decisions are taken by a higher and, presumably, more knowledgeable authority. Either a subordinate human or an automation device assists the supervisor in preparing information, carrying out preliminary calculations, and finding candidate solutions to problems. Having access to additional information, either because of greater training and experience, or freedom from routine tasks, allowing higher-level cognition, the supervisor is assumed to be able to produce more reliable decisions leading to problem resolution. Effective supervision, then, requires that the supervisor be able to add skill, knowledge, or judgment to the information processed by assisting men or machines.

In the coordinated crew training employed by many airlines today, great emphasis is placed on giving the airline captain practice in exercising supervision (reference 22). For example, in simulated flight operations, emergency situations are created that require each crew member to perform an important duty. The captain directs the checklist references, priority decisions, and individual duty assignments. This practice enables the captain to estimate better the capabilities of his flight crew members and to develop skill and confidence in coordinating their activities.

The problem faced by the supervisor, in a more automated decisionmaking and data processing system, is that he may not have the ability to adequately evaluate the "help" he is being given through the automatic decisionmaking capability of the computer, so that he can implement the optimum mode of problem resolution at the proper time.

With the computer not only performing routine calculations but also assisting in problem solving by setting priorities for action, selecting appropriate checklist references, and inhibiting irrelevant information displays, the supervisor may operate at a disadvantage. In effect, the digital system proposes a solution based on preselected logic, and the captain can accept it or reject it. What he cannot do is process all the information himself and understand why the proposed solution is or is not the optimum choice.

The reason for this is that the computer can be programmed to make calculations and assessments that exceed the abilities of the human supervisor. From this, it follows that the pilot may not know all the factors the computer has selected for attention and which have been rejected. The appearance given to an observer might be that the pilot has only low workload in this situation. He merely orders the complex computer decisions to be implemented. What is unseen, however, may be an extremely high mental workload through which the pilot is seeking assurance to enable him to accept or reject or reach his own decision. It is possible that the integrated flight displays and EICAS may be so configured that the needed confirmatory information is provided in a timely and comprehensible form. However, this must be determined by test since it is clear that supervision aided with automated decisionmaking is different from supervision with another crew member possessing like perceptual and calculation capacity and equal access to raw data on the environment.

Data entry by keypack or other machine data entry device is another human role with heightened importance in the computer aided cockpit. As we know from the enormous facility developed by typists, telegraphers, piano players, and accounting machine operators, humans may become rapid and reliable at key pressing tasks. Several general principles have emerged from studies of this class of behavior that relate to pilot workload. First, standardization is of great importance. Workload is reduced if data entry tasks follow uniform conventions of layout, such as the arrangement of alphanumeric keys, adherence to human engineering conventions as to direction of increase-decrease, on-off, etc., and operating feel. In addition, reliability is largely determined by the availability and quality of information feedback. It has been shown that navigation blunders trace mainly to incorrect insertion of waypoint data (reference 23). Such errors, in turn, appear related to the small size of the characters the pilot must discriminate. It has also been found that workload will be reduced if the pilot performs data entry in meaningful groups such that attention can be devoted to each group of entries as an entity. A situation requiring the pilot to insert part of a message, divert attention to a

different task, and later return to resume the incomplete task, would be expected to produce more errors and require more mental effort to complete the task.

Changes in the usual use of speech communication can also affect workload. The most common change with automation is that information previously obtained by listening to natural speech must be obtained by reading a visual display. Since the pilot is generally more loaded in the visual channel than the auditory, such a change in modality serves as an example of the point that mental workload elements are not necessarily additive in the elementary sense. If a channel is fully loaded, workload is not reduced to an equivalent degree if another channel is unburdened. In recognition of the potential utility of spoken messages, the new EICAS provides for the capability to issue voice advisories in addition to those originating in the ground proximity warning system. The effectiveness of this procedure to reduce channel demands will require test; however, certainly there is a potential gain.

6.4.4 Human Needs and Satisfaction.

Psychological studies of pilots have revealed that workload and pilot satisfaction have a complex relation. Few pilots may wish to return to fully manual flight, but many persons prefer to be busy when working and prefer to be physically active as well as mentally involved. For example, a study of pilot preferences showed that the greatest satisfaction followed from execution of a manual approach and landing under restricted weather conditions (reference 4). Stress workload may be high while monitoring a coupled approach, despite a low level of observable pilot actions. The opportunity to demonstrate skill and mastery of difficult tasks may lead to personal pride and confidence in ability to master future challenges. For this reason, it has been suggested that, as automation advances, consideration must be given to meeting pilot psychological needs by "designing in" appropriate activities. Two such activities proposed for consideration by aviation psychologists are increased feedback of operating efficiency data (whereby the pilot would obtain a score or scores showing how near to optimum the fuel consumption rate and other desirable goals were being met), and a built-in flight simulator practice facility. Using the latter during long en route phases of flight, pilots could be provided a simulated landing scenario/task with feedback of manual tracking accuracy (reference 24).

As this general subject has been put by one authority:

"Men differ significantly from machines. When a man is forced to function like a machine, he realizes that he is being used inefficiently and he experiences it as being used stupidly, and men cannot tolerate such stupidity. Overtly or covertly, men resist and rebel against it. Nothing could be more inefficient and self-defeating in the long run than the construction of man-machine systems which cause the human components in the system to rebel against the system" (reference 25).

To summarize the overall impact on workload of automation, it may be said that increased cockpit automation has produced both predictable and well understood favorable impacts on crew workload, and has had the effect of changing the pilot's tasks in ways that may reduce the reliability of his performance. The simplest way to summarize such unfavorable potential impacts is to say that computer automation can effectively replace human effort, in whole or in part, in quantitative roles better than in qualitative ones. Where numerical information is needed, computed

data usually are more precise and timely than manual data. But information conveyed by natural voice, other sounds, human tones and gestures, and pace of activity is lost when mechanized. Such loss of qualitative information may increase mental workload by reducing assurance in the reality or validity of data.

The design of more automated flight deck systems must be sensitive to these human factors issues, going beyond the conventional "knobs and dials" level of analysis. Design proving, then, involves determining that the design flight crew cannot only tolerate the automatic features, but that this every day environment does not impact the aircrew in ways that prejudice the ability of the crew to operate in unusual situations requiring the exercise of manual skills and decision making.

6.5 FUTURE WORKLOAD MEASUREMENT PROBLEMS.

Not only will the pilot role undergo change when computer based integrated avionics systems are substituted for the conventional manual systems, but the workload evaluation techniques most appropriate to the changed pilot roles may have to be different from those used with the previous levels of automation. Performance measures, based on observation of pilot hand and eye movements, may be less appropriate for measurement of monitoring and supervision (decisionmaking) situations associated with the new integrated digital systems, and commonly used subjective measures obtained in post-flight questionnaires may also fail to adequately reflect the workload experienced by the pilot.

Upon completion of a flight, the pilot may not necessarily retain a clear, quantitative memory of how difficult and stressful mental workload events, occurring earlier, had been. Problems solved tend to be problems forgotten (or at least relieved of the strong feelings that may have occurred prior to solution). Hence, it may be increasingly important to use a technique such as that developed by Sheridan and Simpson at Massachusetts Institute of Technology (MIT) (reference 6). At MIT, it was found that more reliable subjective ratings of workload could be obtained if the pilot relived the critical phases of solving emergency conditions by watching a video recording of his own flight deck performance and the instrument indications that were present during that performance. In effect, the pilot first experienced the workload situation and then observed and critiqued his own activities after the fact while reviewing the video. In this way, the workload stresses associated with the ongoing events were recalled, and the pilot was able to rate the scenario events in a way more congruent with other pilots and observers. To date, this experimental technique has not been applied in actual new aircraft tests.

In practice, the procedure more commonly has been to have a debriefing session after completing all the test flights on a given day and to ask the test pilots to sit down and rate workload factors, from pure recall, for all flights, simultaneously.

6.5.1 Evaluation of Stress.

While visual and manipulative events associated with conventional avionics and control systems lend themselves to a time-and-motion definition, the pilot tasks of interest in the 1980's based on integrated computer augmented decision making with the computer aiding the pilot in decision making and supervision are more difficult to measure and quantify. As Sheridan and Simpson point out, this is particularly true of the large transient workload demands that may occur, for example, as a

rather dull autopilot monitoring role suddenly becomes a frantic effort to take control of an unstabilized aircraft due to an abrupt or an insidious/latent failure.

A conventional information flow diagram for the pilot-aircraft-environment loop is shown in figure 6-2. What needs to be added to cover the emotional or stress workload is a notation that the solid lines and arrows may, in fact, be dashed lines (indicating imperfect information or variable errors in display) and the arrows may be double or triple (indicating parallel information flows that may contradict each other or raise incompatible task perceptions). If information always flowed in solid, unequivocal form, and always in one direction with no reverberating question loops, the pilot would have little chance for error and stress would be expected to be low. Cockpits are designed to maximize this probability. However, when system failures or other untoward events occur, mental workload ceases to be a function of simple, sequential intake of solid information, cognitive processing, and control input activity. It is here that stress and emotional workload come into play, and decision making becomes complex due to the potential results from erroneous decisions and inputs.

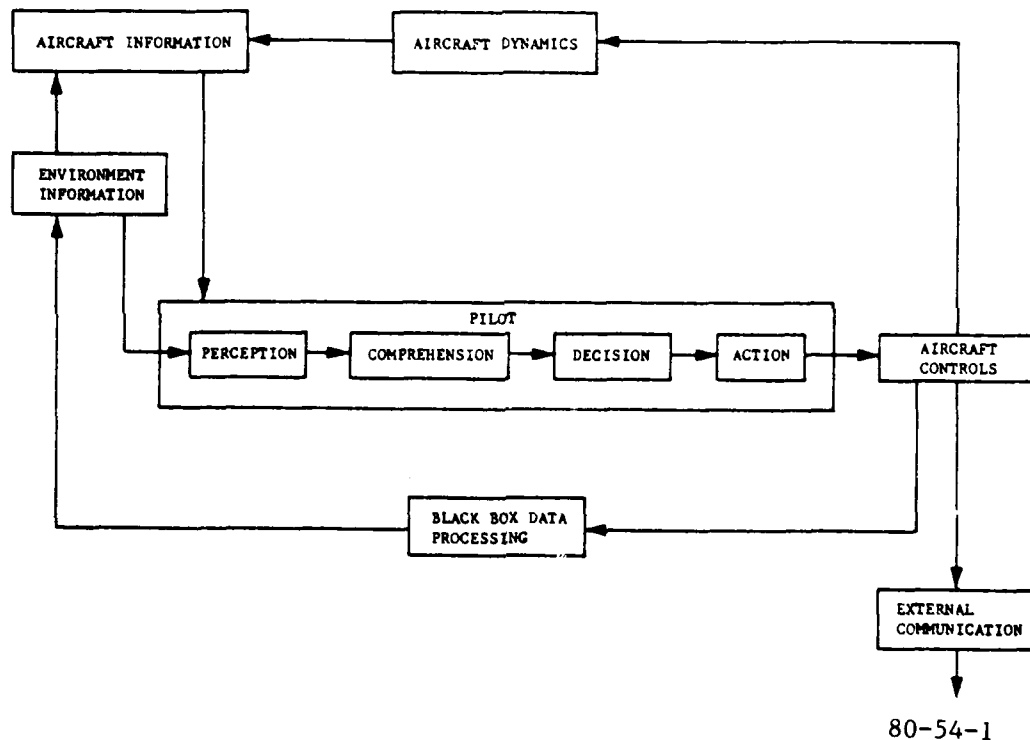


FIGURE 6-2. THE PILOT-AIRCRAFT-ENVIRONMENT INFORMATION FLOW

The stress aspects of mental workload occur as a result of the pilot's mental and cognitive processes and often are not directly observable to the workload assessor. There are likely to be physiological correlates of mental stress that can be detected by more subtle measurement techniques. However, these types of measures have not been used on any routine basis in past transport aircraft workload programs nor have the appropriate measurement activities been defined. It is suggested here that as flight decks become more demanding in terms of monitoring,

supervision, and decision making activities, and less demanding in physical activities that can be directly seen by an observer or camera, some shift in workload measurement technology toward the physiological measurement options may become necessary and, in fact, mandatory in order to determine the actual workload/stress.

6.5.2 Sources of Pilot Error.

The whole purpose of evaluating flight deck design and assuring acceptable pilot workload is to ensure proper pilot performance and to reduce the probability of pilot error. The ways in which pilot errors may occur and the links between these potential problems and the design aspects of flight deck development and implementation may be categorized with the idea that a checklist of good design points and another list of potential design pitfalls may be constructed.

Several extensive studies of aircraft accidents, and particularly turbojet transport accidents, have been published in recent years (references 26 and 27). Many accidents are one-of-a-kind situations, and others result from extremely improbable events external to the pilot-aircraft control loop, but there remain several classes of accidents of interest. One of the most important is controlled flight into the terrain. Another is improperly executed instrument approach to landing. Rarely have such accidents been traced to features of flight deck design or other instances of improper engineering of the pilot/machine interface. A few examples of this kind do exist, however, the case of the initial DC-8 lift-spoiler system has often been cited. Several accidents occurred after lift was lost when spoilers were deployed prior to touchdown. Since some aircraft controls are arranged so that there is a normal in-flight armed and deployed position for spoilers, while there is not a correct parallel use in the DC-8, a potential pitfall existed for pilots accustomed to the other aircraft. The corrective action adopted for DC-8 spoiler deployment was the installation of a guard on the switch. The general principle involved is that design should make inadvertent actuation difficult or impossible if the function in question is safety related. Related examples of this design consideration include the Los Angeles B-727 spatial disorientation accident. Investigation showed that the nonstandard electrical control panel placed galley power cutoffs next to the essential battery power switch. With a generator failure, the power to flight instruments may have been interrupted when an attempt was made to shut down nonessential power.

Returning to the pilot information processing loop illustrated in figure 6-2, examination of the stages of pilot-aircraft interaction in the cockpit may suggest major categories of possible pilot errors:

1. The pilot in the cockpit is viewed as engaged in "perception" of information about his aircraft state and dynamics and information about the environment. He senses and detects significant information which may include, of course, a danger signal. This detection of significant information may be impeded if an important signal is not in the pilot's field of view, is blocked, or is not discriminated from the noise context because it is too weak in signal strength or is unclear.

2. The term "comprehension" is meant to suggest the recognition and interpretation of the importance of incoming information. A signal might be perceived, for example, but not be recognized as having an important bearing on some aspect of the

situation that is, in fact, critical. The complexity of displays and the magnitude of the workload imposed by concomitant cockpit tasks may influence the completeness of comprehension.

3. "Decision" is the information-processing phase in which the pilot selects, from a repertory of alternatives, the particular action that is appropriate. A danger signal may be perceived and its importance may be comprehended, but the correct action may not be

4. Finally, a failure may occur when the pilot implements the selected action. The physical action itself may be poorly coordinated or incorrectly performed, for example.

Each of the four areas of pilot information processing in the cockpit — perception, comprehension, decision, and action — can be affected by the design and operation of cockpit systems. Standard instruments, long familiar to the pilot, and standard usage of coded knobs and dials will increase the probability of perception. Signals arranged in a standardized array and received without excess demands competing for pilot attention will be best comprehended. Errors in decisionmaking may stem from cockpit systems that are more complex or attention-demanding than is necessary. Finally, an action selection failure may be promoted by cockpit arrangements that make it possible to confuse one control with another, so that the pilot who means to do one thing actually does something else. Standard arrangements and logic of actuation are clearly among the means of reducing such action failures (reference 28).

6.5.3 Definitions of Good Design.

In this discussion, phrases such as "well designed" or "human engineered," referring to cockpit systems, are used as shorthand to refer to displays, controls, cockpit layouts, and auxillary systems that conform to that which accepted as good engineering practice and judgment. There are general principles that enable one to determine whether a cockpit system is well designed, and these general guidelines should be stated.

Standardization is, itself, of very great importance in any complex task that is performed on the basis of past training and experience with similar or analogous systems. An everyday example is found in the typewriter keyboard. Even a beginning student of touch typing can determine that the standard "QWERTY" keyboard layout is far from optimum. It does not spread the workload equitably among the fingers, but standardization is of such overwhelming importance in a skill such as typing that we retain the traditional layout. The cockpit of an airplane presents both traditional tasks for which there are well established population stereotypes, constituting old habits that may be relied upon, and also novel displays and controls that have been created specially for the aircraft situation. For each of these, the old and the new, there are generally accepted rules of human engineering that tend to ensure that the system is easy to learn and use, is resistant to serious error, and takes account of the special information processing capacities and error potentials of human pilots.

1. Selecting a cockpit system for an "old" or traditional task (such as directional control, power setting, on/off selection), the paramount considerations are as follows:

c. All systems must follow population stereotypes as to logic of actuation, direction of increase, and "natural" relations, such as "turn left" to select the left.

d. Positive detents or other provisions to prevent inadvertent actuation must be provided on all hazard controls.

e. Provision should be made for continuous or instantaneous testing the status of systems, and indicators should have a clear failed-state.

f. Standardization should cover, where appropriate, the location, size, color coding, shape, labeling, feel, logic, and arrangement of related systems for all important devices and systems.

2. In the case of a novel aircraft system without a common analog in the experience of most nonpilots (such as readout of radar, altitude, or instantaneous vertical speed, flight director symbology, or a control for setting cabin altitude differential), a set of more general design guides may be stated:

a. The design should be based on a systematic study of both the purpose of the device and when and how the pilot uses it.

b. Information processing sequences should be considered so that there is the maximum distinctiveness and separation of confusable but incompatible systems.

c. Simplicity of display and action should be sought, recognizing that the system may have to be used in conditions of excess workload or "panic" (quick reaction) situations.

d. The "perceptual strength" of the human in recognizing patterns of information should be considered in display design.

e. The "response limitations" of the human should be considered in design so that the pilot is not required to perform difficult and demanding coordinations in any of the sensory or physical channels.

f. Planning aids and feedback from responses should be included.

6.5.4 Relation of Workload to Safety.

In Wiener's analysis of accidents (reference 26), failures of information processing are seen as system-induced errors, most of which are preventable by judicious consideration of human factors. Obviously, high workload may contribute to such failure to perceive and act on available information, particularly if the flight crew elects to continue an approach or other critical maneuver in the face of multiple problems.

Delta Flight 727, A DC-9, crashed short of the runway at Boston in 1973 (references 26 and 29). The National Transportation Safety Board (NTSB) determined that the probable cause was failure to monitor altitude and recognize passage of decision height during an unstabilized approach. The aircraft had passed the outer marker above the glideslope and at excessive airspeed, resulting in part from unstandard ATC services. In addition, the flight crew workload may have been increased by unstable and questionable information presented on the flight director.

High workload is characteristic of an unstabilized approach, and preoccupation with cross-checking questionable computed Instrument Landing System (ILS) information possibly could have diverted attention from a normal instrument scan. The most striking aspect of Wiener's analysis of this and other controlled flight into terrain accidents, however, is that such high workload is not generally found. Crew inattention without pressing demands for other crew work was more common. NTSB accident summaries were found to show a preponderance of phrases such as "lack of vigilance," "neglect of flight instruments," "failure to monitor altitude," "deliberate descent below flight minimum," etc.

One hypothesis, advanced by Wiener, to explain such pilot behavior is that automation improvements are making flying too easy, that over-automation of control functions leading to pilot underload has led to complacency, lack of vigilance, and errors during those critical times when human judgment and intervention are required immediately in order to correct or salvage some problem area. Pilot workload then may be related to safety in the form of a curve as illustrated in figure 6-3. Safety rises as workload rises to a medium level supporting vigilance, alertness, and knowledge of the ongoing situation. Further, large increases in workload may lead to ultimate inability to process all the information, uncertainty, high stress, and a safety decrement. This concept includes not only the quantitative aspect of workload — which might state, for example, that 25 percent to 50 percent busy is best for safety — but also the qualitative aspect. Requiring irrelevant "busy" work of the pilot would not necessarily advance his knowledge of the ongoing situation. Additional workload consisting of information processing in the environment-pilot-aircraft control loop may enhance pilot capability to assess new information or handle warnings/critical situations.

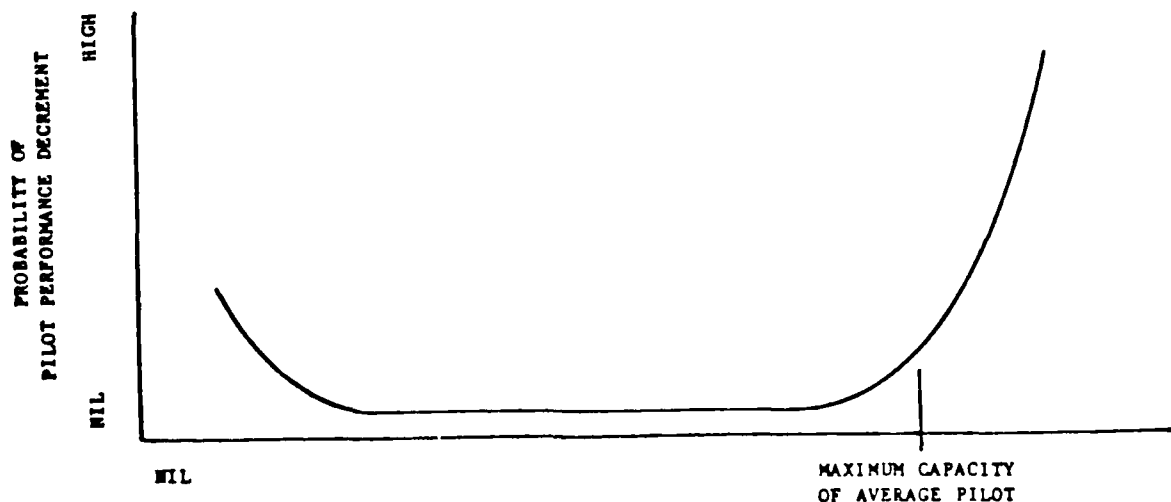


FIGURE 6-3. RELATION OF HIGH AND LOW PILOT WORKLOAD AND SAFETY

Evidence exists that too high workload increases the probability of pilot errors. In a full mission simulation test, a dramatic increase in errors, from 106 to a total of 292 in the combined areas of navigation, communication, system operation, flying tactical decision, crew interaction, and flying skill, occurred when high workload was compared to low. Errors using the autopilot were, however, higher in low workload (reference 30).

The optimum procedures for assessing pilot workload should provide for test of both the quantitative and qualitative aspects of crew workload. As automation implementation proceeds in new and derivative aircraft, it may be easy to produce measures showing that the pilot has neither an increased number of tasks to complete nor more difficult task demands than were experienced in comparison, service-proven aircraft. A considerably more challenging problem may be presented by the need to develop measures that show that the new integrated systems do not promote the underloading phenomena that have already been identified by some analysts as detrimental to vigilance and safety. The companion problem, on the qualitative side, will be to assure that the pilot is not being forced to work as a low-knowledge machine supervisor and data entry clerk. The proper level of monitoring/vigilance must be identified and maintained by appropriate built-in tasks and other secondary activities which insure the proper level of both physical and mental activity during the entire mission. These ancilliary tasks must be of such a nature that they do not preclude the pilot's ability to monitor and recognize cautionary and warning trends or activities and in fact do not establish a perceptual set or modify appropriate behavior by causing the pilot to be more interested in the outcomes of the ancilliary task than the primary task.

6.6 RECENT CERTIFICATION PROGRAMS OVERVIEW.

A study team sponsored by the Wright-State University has made a detailed review of flight crew member workload evaluation techniques (see reference 31). This review included the certification programs conducted for the most common two-pilot airlines aircraft, the DC-9 and B-737, as well as those conducted for the most widely used of all transport aircraft, the B-727, and later wide-body aircraft, the DC-10 and B-747. Hence, a comprehensive list of previously used and successful methods of demonstrating crew workload was assembled, and present reference may be made to that inventory.

6.6.1 Systems Analysis.

In time sequence, one of the earliest methods of comparing crew workload between a new aircraft and an existing design is a systems analysis. This may be initiated by compiling checklists and required procedures lists for those aircraft. Checklists are required for commercial aircraft by FAR 121.315 (reference 32) and include enumeration of each item the crew must verify for safety in engine and system emergencies and in each normal transition mode; e.g., pre-taxi, takeoff, climb, descent, pre-landing, etc. Procedures lists are more detailed explanations of crew duties required for operation of the aircraft.

In the B-737 systems analysis, checklists and procedures for the new aircraft were compiled and compared to three existing inservice aircraft: The DC-9, BAC-1-11, and B-727. Except for the new aircraft, the lists and procedures were taken from the manufacturer's operation manuals to ensure completeness. All of this material was organized under four headings:

1. Normal checklist (taxi check, before takeoff, after takeoff, descent, approach and landing).
2. Procedures-system operation normal (fuel, miscellaneous, flight controls, powerplant, hydraulics, electrical, APU, heat/anti-ice, etc.).
3. Abnormal procedures (hydraulic pressure low, loss of System A, loss of System B, loss of both systems, pump overheat, etc.).

4. Emergency checklists (engine fire, engine overheat, tail compartment high temperature, etc.).

A count could then be made to verify that the number of items was much less than that of the three-crew B-727 and comparable to the loading on the Captain and First Officer of each aircraft already in service. A count of this kind is not considered to be particularly useful since the extent and quality of workload varies too much from item to item. Only insofar as the items are highly comparable would a raw count represent a workload difference.

A major limitation of checklist comparisons is that these lists are focused on subsystem operation, the things that are done rarely, so they may be forgotten unless a mechanized procedure of challenge and response is followed. Checklists do not cover the major pilot activities of flying the aircraft, dealing with air traffic control, and updating avionics for navigation. Much of pilot workload is omitted. The major attraction of checklist study is that it often is just in subsystem operation that workload reduction is sought. Checklists are provided for important contingency conditions as well as for such normal operations at initiation of each phase of flight. To the extent that workload peaks are associated with emergencies, such as engine failure, evidence of reduction in the number and complexity of required checklist steps might be important.

Crew Procedures Objectives (CPO) represent expansion on the concept of checklists and are a later development from the Specific Behavioral Objectives (SBO) emphasized in recent years. CPO's are developed through a cooperative effort of the manufacturer and customer airlines. The principal purposes (of CPO) are to guide the training of pilots and to produce standardization in accomplishment of assigned tasks that ensures that any qualified pilot on the company roster can function cooperatively and knowledgeably with any other company pilot as a disciplined team member. Thus, CPO's are not part of certification; the airline operating the particular aircraft can change them as appropriate. The activities related to development of an initial, industry-standard set of CPO's for a new aircraft are such, however, that a clearer understanding of task demands results. If there should be features in the new flight-deck configuration that cause special workload problems, these features would be highlighted in the CPO documentation process and would be expected to come into general knowledge; in addition, it is hoped that the manufacturers would initiate an activity to reduce the apparent workload by use of alternatives or by changing the presentation or the level of augmentation by computer, (i.e., increased automation).

A minimum equipment list (MEL) comparison will usually be made as well (reference 31). FAR 121.627(c) permits the publication of a MEL designed to provide operators with the authority to operate an airplane with certain items or components inoperative, provided an acceptable level of safety is maintained. The list does not include obvious required items such as wings, flaps, engines, etc., and it also may not include items unrelated to the airworthiness of the aircraft, such as galley equipment and passenger convenience items. The list does include items related to airworthiness which may be inoperative for airplane operation, and the limitations and use of other operating components required when dispatching with inoperative equipment (i.e., CAT II, III exclusions; VMC only, etc.).

A MEL comparison was made when the B-737 was being developed. A count was made of the MEL items that might cause major, minor, insignificant, or no increases in workload for the new aircraft, with its several simplifications, and for a

comparison aircraft, the B-727. The obtained information was discussed with particular consideration devoted to the questions: what role would a third crew member have in dealing with inoperative equipment problems; what duties could he perform; and what could he reach and see? In a somewhat similar manner, for the initial model for the DC-9, an analysis was presented based on the quantitative count of MEL items which resulted in reductions in required procedures, along with an estimate of the workload impact for each advance in automation/simplification, and documentation which illustrated the limitations in external visibility from an added rear seat (i.e., third crew member position).

Historically, MEL has not been a required subject in crew complement certification. FAR 25.1523 and appendix D (reference 1) do require examination of abnormal procedures and emergencies, but make no mention of MEL. As aircraft are designed to rely more and more on computers and computer-based digital systems (particularly in the case of relaxation of natural aerodynamic stability intended to reduce fuel consumption), it may be that the importance and criticality of MEL determinations is increased, unless redundancy, including analytical observers, is included at all levels and is sufficient to assure that fail-passive/fail-operational goals are met. The potential dependence of crew workload on the status of optional automation systems suggests that, in the future, it may be necessary to determine crew complement and the MEL in certification, and to do this with at least a "target" set of CPO's, showing the division of responsibilities and required actions among the individual crew members.

6.6.2 Synthetic Performance Measures.

Soon after completion of the detailed design of the new aircraft flight-deck systems and crew procedures, it is usual to configure a simulator (or initially a mock-up is established) to provide a realistic model of the new flight deck. This is usually an advanced developmental cab which may or may not be identical in all dimensions and secondary features to the new design; however, it is computer driven so that essential flight instrumentation and engine and systems indications are dynamic and function as intended in the design aircraft. The checklists and procedures compiled in the systems analysis phase can then be used to generate actual pilot performance data which is usually predictive of workload in the actual aircraft. Normally, the goal of this test effort will be to show that workload is acceptable to the pilots. The term "acceptable" is a broad term that may cover several differences as follows: a better balance of workload between crew members; a reduction in peak workloads encountered in the reference aircraft; or a general reduction in subsystem monitoring and adjustment requirements, while actual flight control activities remain essentially standard. Since the general consensus among workload specialists holds that basic flight workload in current transport aircraft is acceptable, special simulation and task/time study activities performed after aircraft design may, alternatively, have the goal of demonstrating to the FAA that workload demands on the crew, and individual pilot workloads, are equivalent to, or no more extensive than, those in reference aircraft that have been proven acceptable in actual airline service.

Comparable studies performed by the manufacturer prior to the essential completion of flight deck design were, of necessity, limited to part-task simulations. At this stage, however, there exists a complete enough description of the future flight deck to allow the manufacturer to initiate studies that approach the level of full-mission simulation. The data that are needed for such a demonstration of overall acceptability of the new design must be specific with respect to the crew seat, the type of flight operation, and perhaps most important, the comparison or reference aircraft.

(NOTE: It might be good at this time to point out that, in the lead certification regions, up to the present time there have been no human factors specialists on the staff or as Designated Engineer Representative (DER's). This may be changed with the establishment of the Certification Directorates and the National Resource Specialists.)

Each system included in the flight check design is evaluated as to failure modes, indication of failure, recommended response to failure indication, and consequence of failure to respond to a warning. This is to assure that the crew will not have to provide constant attention to a system, and that system controls and indicators have been given a priority of location consistent with the importance of the particular system operation under normal and abnormal condition. Multiple malfunctions that could compound abnormal and emergency functions, such as the loss of a single-engine and the opposite engine generator on a single flight, are evaluated. The accessibility and conspicuity of all necessary instruments and controls are evaluated. Evaluations are made under bright sun light (simulated with high intensity lamps), normal lighting, and medium to full darkness. Studies that are usually conducted include peripheral vision of instruments, visibility and readability of instruments, and capability of each pilot to see and reach required controls. Pilots ranging in height sufficiently to represent the prospective pilot population are used in the study.

In most of the recent certification programs, pilot performance has been translated into one of several numerical indices or probability distributions to facilitate comparison with studies on other aircraft designs. Workload may be expressed in terms of a time-line or set of time-lines which show the value or extent of the workload (expressed as a percentage) at any instant in a given mission. These also show multiple activities which are concurrent and give a total estimate of percentage of available time. It should be kept in mind, however, that as skill increases in working in a complex situation, the operator or pilot finds that he may not have to think, or devote all of his attention, to any one action; in other words, he reacts instinctively. The pilot can, in fact, look outside and receive information at the same time that he is making control movements or receiving aural information.

The assumptions of single channel capacity in mental workload employed in numerical combinations of time durations for required actions or time probabilities are, therefore, largely artificial. Hence, the absolute numbers obtained do not predict real flight workloads; generally, they overestimate. For this reason, the obtained numerical workload levels are useful primarily for comparison with similarly obtained numbers on a reference aircraft. Past experience shows that properly conducted synthetic studies can predict equivalence, or the direction of a difference, and trends despite the limited meaning of the absolute numbers themselves.

Pilot performance measures obtained in test series in the ground simulator may be replicated in computer simulation to provide information as to the impact of a wide variety of weather, ATC, and system operation variables. In addition, manufacturers of FAR Part 25 approved aircraft have developed, and used successfully, various other computer modeling studies. Geometric data are used to evaluate the visibility and accessibility of items in the cockpit prior to final construction of the aircraft. Angular movements and changes in linear distances for the eyes and hands as they move to perform aircraft procedural tasks during flight are calculated. Combining these data with programs covering task sequences allows determination of angles from the eye to points of interest and fields of view

during crew-member procedural activity. Changes in angles and linear distance can be summarized for mission segments, showing how much eye fixation must be shifted to operate the aircraft. Eye/hand coordinated motions and procedure execution times are included in various programs of this sort so that overall workload results from arrangement and procedure changes may be evaluated quantitatively. Accessibility-reachability is tested mostly in the three-dimensional mock-up. Available analytic tools such as computer modeling methods are being improved but are considered less useful than live demonstrations at this point.

External vision studies are made to determine the amount of outside visual field available for the pilots compared to proven, in-service aircraft. The available visibility that a crew member has is a function of the shape and size of the window and the eye distance to the window. In engineering terms, this is measured in steradians or the solid angle subtended. From the external vision measurements, charts can be developed to document that the new flight deck design meets reasonable criteria of external visibility. Once again, the primary utility of all of these pilot performance measures is to make a comparison of crew member workload with that of a reference aircraft.

The obtained data from the modeling of pilot performance measures, whether reported as a straight workload index reflecting percentage of available time occupied with crew-equipment tasks, or in the alternative task/time probability distribution format, presents as the most important question: How should the results be interpreted? If the calculations result in peak workloads higher than those in critical flight scenarios for reference aircraft, it might be thought that the answer is simple — design will not be acceptable. Several caveats must be stated on this point, however, since not all peaks are so high as to represent genuine problems, and there may be compensatory gain. For example, in one DC-9-80 study, it was found that the first officer's workload increased in one phase of flight compared to the baseline aircraft (reference 17). The peak increase was judged to be acceptable, since a better balance between captain's and first officer's workload was achieved, and in this particular instance, it was that balance, with reduced demand on the captain, that was most significant (figure 6-4). There is, furthermore, some reason to emphasize average workloads, over reasonably brief periods, as having more meaning than essentially momentary peaks. The reasoning here is that despite CPO's and standard training, no two crew members will conduct required tasks in exactly the same sequence or with the same response latencies. Pilots recognize busy periods and adjust priorities and adopt task interleaving techniques and strategies to manage high momentary demands and minimize peaks. Because of this built-in human capacity for flexible response, the data-base task duration times, while appropriate for average situations of normal demand, may tend to overestimate the required task time under higher demand pressure.

Aside from a possible finding of peak workload in excess of that in the comparison aircraft, under circumstances such as those just covered, the typical finding has been that the resulting calculations for the new aircraft do show some workload reductions for particular segments and rough equivalence to reference aircraft in others. How, then, should this outcome be interpreted?

There is no general agreement that the optimum workload for aircraft crew members is an absolute figure (i.e., 20, 30, 40 percent). The nature of the activities requiring allocation of a given percentage of available time, the difficulty of required mental efforts, and other considerations such as emotional stress, physical discomfort, professional pride and social satisfactions in a team job well

done, versus feelings of competition or hostility, will make a difference in any person's perception of how much work is enough and how much is too much. Thus, it is extremely difficult to determine what weight to place on obtained differences in a simple workload index for a specific aircraft configuration.

While authorities agree that total workload cannot be measured in such a way that a numerical safety factor can be derived, the various workload measures obtained in simulation, computer, and analytic studies can be compared between a new flight deck design and proven, inservice designs. No single comparison, as a simulator study alone or a task/time probability distribution alone, is considered to provide an adequate comparative evaluation. Taken together, however, the results of the several types of studies have proven to be predictive of the real-world workload effects.

	CAPTAIN	FIRST OFFICER
TAKEOFF WI (-50)	49.79	49.12
WI (-80)	33.57	36.57
% WORKLOAD CHANGE	32.8	25.6
CLIMB WI (-50)	29.10	25.28
WI (-80)	18.31	23.08
% WORKLOAD CHANGE	37.1	8.7
CRUISE WI (-50)	21.63	16.85
WI (-80)	17.52	16.08
% WORKLOAD CHANGE	19.0	4.6
DESCENT WI (-50)	35.29	26.78
WI (-80)	26.43	25.16
% WORKLOAD CHANGE	25.1	6.0
APPROACH WI (-50)	38.72	29.77
WI (-80)	31.04	26.06
% WORKLOAD CHANGE	19.8	12.5
LANDING WI (-50)	54.25	18.93
WI (-80)	40.33	22.45
% WORKLOAD CHANGE	25.7	-18.6
OVERALL WI (-50)	26.44	20.92
WI (-80)	20.49	19.62
TOTAL % WORKLOAD CHANGE	22.5	6.2

NOTE:

NEGATIVE NUMBER = DC-9-50 WORKLOAD LESS THAN DC-9 SUPER 80

FIGURE 6-4. DC 9-80 COMPARISON OF EQUIPMENT INTERFACE WORKLOAD WITH DC 9-50 WHEN AUTOTHROTTLES ARE INOPERATIVE

In the face of these difficulties of interpretation, the nearly universally adopted solution is to compare measured or calculated workload during what are deemed to be the most critical minutes of a total cycle, usually the phase of final approach and landing. This is not to imply that workloads are not calculated for all phases of flight or that attention is not given to taxi, takeoff, climb, en route, and descent phases. Rather, the point is that the most critical demonstration of workload suitability may be made for final approach and landing. The reasonable assumption is made that the individual crew members in proven inservice

aircraft are capable of performing these critical operations--as documented by the safe landing records of the comparison aircraft. Therefore, the task demands in those aircraft must not be too high (and any observer would probably verify that during approach and landing they are not too low to sustain performance). If the measurements and calculations in the new design show that the individual crew members have equivalent or lower workload results during similar scenarios, it is generally concluded that the tests have shown workload to be acceptable. Equivalence is the key. Workload must compare favorably with another "good" airplane.

Hence, there is no penalty, preventing the giving of credibility to the data, for lack of an absolute scale for optimum and obtained workloads. Comparison provides something more substantial than would such an arbitrary, and arguable scale. What this comparison standard is, really, is a summary fact based on many thousands of previous flights, accomplished by a wide diversity of pilots, under highly varied conditions of route, weather, traffic, and ATC. No known synthetic or laboratory procedure could possibly produce pilot performance data of this generality.

6.6.3 Flight Test.

Initial production aircraft are first test flown by the manufacturer's pilots, but very soon direct FAA participation begins, and a program of functional and reliability (F&R) test flights is prepared. These tests are concerned primarily with aircraft performance and systems, rather than crew workload, but an immediate result of the test series is an accumulation of pilot experience in operating the aircraft with its new features and flight deck provisions. The same company and FAA pilots who conduct these test flights may be included in the panel surveyed later to provide pilot opinion data. Ultimately, some portion of the later F&R flight program may be designated for specific test of pilot workload, and in programs with critical changes in flight deck arrangements, a flight simulation of airline operations may be arranged.

As a first step toward conducting a workload flight test, the manufacturer prepares a test plan which provides realistic inservice conditions such that actual workload can be observed, recorded, and reported for both normal and abnormal (contingency) conditions. Critical elements of that plan, which must be approved by the FAA, include: the selection of test pilots, the number of test crews, the selection and preparation of flight crew observers, the data recording methods to be used in flight, the data analysis procedures to be applied during and after the flight, the particular contingency conditions and combinations of conditions to be covered in the flight tests, the number of replications of tests, the route structure, length of duty cycles and stresses to be controlled in the flight environment (including weather and ATC problems), and the criteria of success. In addition to the initiative required of the type certification applicant to prepare the test plan, and the TCB participation that is aimed at ensuring that the plan is adequate in all respects, it may be the case that outside parties will make proposals for additional or special test conditions. All such proposals will be considered jointly by the manufacturer and the TCB, but such outside contributions of ideas and information create no obligation on the part of the TCB to discuss the overall plan or its details with such informants. The sole responsibility for approving the plan, as is the case with all other aspects of airworthiness approval rests with the FAA, under the law.

Flight test is a continuing process, not a phase initiated and completed as part of aircraft certification. The F&R program is such a phase of airworthiness proving,

but represents the culmination of a larger testing activity, not the whole substance of that activity. Ordinarily, new developments are flight tested for periods of about 5 to 8 years before going into a new aircraft program, with such experimental work conducted by a variety of organizations. Examples include Government programs on advanced cockpit displays, such as the TCV aircraft, flight test programs for avionics developments and flight management computers in aircraft operated by major avionics suppliers, and FAA, DOD, and aircraft manufacturer's flight tests of various new developments in navigation, landing, and control systems. This substantial flight development and proving often leads to the development of industry-wide standards, in many cases with promulgation by Airlines Electronic Engineering Committee (AEEC), Society of Automotive Engineers (SAE), Radio Technical Commission for Aeronautics (RTCA), and similar broad-based committees of published "Specifications and Characteristics," issued by Aeronautical Radio Incorporated (ARINC), "Aeronautical Recommended Practices," by SAE, "Technical Standard Orders (TSO)," by federal agencies, and the like. Thus, before a new development is injected into an aircraft program where there would be a risk to the certification applicant if the system proved unsatisfactory in actual flight operations, both extensive flight test and approval of design, manufacturing, and interface criteria normally have been achieved.

In view of the above, certification flight tests do not constitute the entire experimental, comparative, and procedures evolution phases of new flight deck system development. Rather, certification represents a brief but systematic examination of the specific configuration and particulars of the integrated function within that configuration. The component equipment either has been proven in previous aircraft or, if new, has been proven in long-term flight test programs leading to essentially standard rules of design and performance. Certification testing proves that the cockpit systems integration has been done right so that the crew and all systems can work together. Of particular value in providing pilot experience on which subjective judgments of work demands may be based are so-called mini-airline flight test series, examples of which follow. With the new integrated systems, integration is even more important than component testing.

Mini-airline flight tests are used in major airline workload programs. In recent certification programs for two-pilot crew aircraft, such as the DC-9-80, Eastern United States (U.S.) routes including Washington, New York, and Boston have been used with duty-day length and number of operating cycles patterned on the longest and busiest found in any airline schedule. To further increase the potential for fatigue, individual crews have been scheduled to fly the test aircraft on as many as three consecutive days. Concentrated programs of flight test in high density environments have been called "mini-airline" tests because of the high degree of simulation of actual airline conditions.

In the B-737 certification program, heavy traffic, noise abatement procedures, saw-tooth altitude approaches to clear Kennedy traffic, the holding patterns, and short cruise times made these routes ideal for the test. Conducting the test over a full week during the Thanksgiving holiday period, with flights scheduled over a 12-hour period, was selected to assure peak traffic for at least part of the test period. The weather provided both dry, clear days, and cloudy, foggy and wet days for the flights. Furthermore, crew duty times were increased beyond normal to provide opportunity for replication of the most unfavorable pilot schedule conditions.

Fifteen flights of the test were conducted with simulated inoperative equipment covering almost all malfunctions that could significantly contribute to high workload, such as autopilot failure, cabin pressurization failures from auto and standby modes, generator failures, and hydraulic failures. Single pilot flights were conducted from each pilot's seat. The combination of the holiday period, weather, day and night, flying eastern U.S. routes, simulated malfunctions, single pilot operation, and a pilot crew with almost no previous experience on the routes combined to make almost an ideal test bed for two-man crew evaluation.

In the case of the stretched DC-9, model 50, 5 days of daytime flights and 1 day of night flight were conducted in the Yuma and Los Angeles areas, with both instrument and visual meteorological conditions, different types of landing approach (coupled, manual, and circling), and selected contingency conditions (pilot incapacities, MEL items such as generator out, and emergencies such as engine failure on approach, pressurization failure, and unruly passenger). Three crews flew the 8-hour day schedules, representing the manufacturer and the FAA. With the later DC-9-80, an eastern U.S. route segment was selected between Atlanta and Boston.

Contrasting with the "mini-airline" scheduling for high workload flight testing, various recent three-crew aircraft have used schedules that were less structured but nonetheless stringent. The B-747 was taken on a world tour, for example, to examine crew performance under highly varied workload conditions produced by a range of ATC, airport aids, and traffic environment factors. While the B-747 tour preceded certification, the B-727 was taken on a world tour after certification. At the time the B-727 was put through F&R testing, its three-crew flight deck raised few critical workload issues, so the F&R flight test program was pointed to provide a sufficient service experience to establish the reliability of operations with a particular layout of aircraft systems, each of which had been extensively tested in other operations.

There are several requirements for valid flight test. It is important that all test crews have adequate ground school and simulator training in the features, checklists, and flight manual of the test type prior to conduct of data flights. Normally, several of the pilots used in F&R workload tests will have additional experience with the new design gained during the aircraft design process and earlier non-data test flights. Those pilots may also be familiar with the outcome of simulator and computer evaluations of workload using the new flight deck design and procedures. This prior familiarity gives those pilots an advantage in making the in-flight evaluations since they know what especially to look for, what are the presumed advantages of the new design, and which procedures have been changed due to new equipment, simplifications, or increased levels of automation.

In the case of the two-crew aircraft, the usual number of test crews is not less than three teams, each consisting of company and FAA pilots. Each crew is expected to operate the aircraft for a period of time sufficient to reproduce line-service fatigue conditions and also to serve as observers and evaluators of other crews flying on subsequent days. Hence, the "duty" periods of the test crews consist not only of the days that they operate the test aircraft, but added days riding jump seat or observing flight deck activities on video monitors from the cabin.

While it is generally considered unsatisfactory to rely entirely on the pilot's rating of his own performance, it has been shown in past test programs that a trained observer who knows the pitfalls of self-rating can note much that is missed by the self-rater. The trained observer can fault a performance even though the

actual crew would give it a success rating. The priority, then, must be put on the assessments made by "senior" observers playing the part of an evaluator. To support their analysis of the flight events, video playback of flight deck signals and pilot actions has proved of great value. An overall "good" rating covering an 8-hour flight with various departure cycles would be expected to be less valuable than a phase-by-phase or contingency-by-contingency evaluation made after observing the entire flight and then replaying the records of various high workload phases.

Discussions among several such trained observers while replaying the pictures and questioning each other as to the reasons that the crew acted as they did can be valuable in adding insight into the ratings.

In the B-737 workload program, video was used primarily to present a picture of flight deck events to observers situated in the rear cabin making real-time evaluations. In the later DC-9 programs, recordings were made with the principal intent being later review and analysis. When there is a particular cockpit feature that is changed or a new system that is added, video tape may be employed specifically to record the use and performance of that feature. Somewhat related procedures were used in the B-747 program. Tests pilots asked for review of specific flight deck procedures after seeing a film that was made using a mock-up using all of the normal procedures. With the mock-up, these pilots then reviewed the questioned procedures and repeated the necessary actions in live action to recheck accessibility, visibility, and related concerns. Also, on the B-747, what was called a "Substantiation of Analysis by Live Demonstration" was conducted by putting the pilot through a structured simulator flight with tape pacing of events. This constituted one form of a "live" time-line.

Objective data to be recorded during test flights normally includes aircraft performance parameters such as altitude, speed, bank angles, and the like, the ATC record of clearance changes and other communications, and the actually attained schedule of planned contingency events plus such unplanned outages and problems as may have occurred.

Using the method of Douglas Aircraft Company (reference 17), analysis of tapes made during flight takes 1-1 1/2 hours coding time for each minute of actual flight. The result of that labor-intensive study of small segments of crew activity is a detailed record of every observable crewmember action arrayed in a chronological sequence showing when the event began, how long it lasted, and how often each action was repeated. An objective record of this kind, representing, for example, a segment of 10 minutes of an arrival in a high density environment, is used in two ways. First, it gives the trained observer a solid basis to reconstruct the pilot reactions and altered procedures that were forced by the contingency conditions employed. This tends to ensure that the observers' ratings are as "objective" as possible. Second, the record based on events during actual flight can be compared to the similar workload estimates generated in simulation. This comparison may permit an estimate of the validity of the measures that were based on computer models or part-task simulations.

From time to time, criticism has been voiced about substitution of the one readily available method of assessing total pilot workload and comparing it to that of another aircraft, pilot "subjective" evaluation, for the more quantitative and methodologically desirable "objective" measurements. Workload authorities make two points related to this criticism. First, the history of past workload determinations by pilot assessment shows that the correlation between the informed judgments

of qualified pilots, on the one hand, and the proof obtained in extended periods of diversified line experience, on the other, is extremely high. Second, there is no other comparable human team performance that can be rated in toto by so-called "objective" methods. It is a fact of life that there are strict limits to what we can measure by directly instrumented procedures, and these limits never include all important cognitive activities and emotional experiences such as creative problem solutions, subjectively experienced stress, and both reactive and "free-floating" anxiety. Hence, it may be foolish to overstress the need for objective measures per se. Rather, the goal that is more appropriate may be to "objectify" pilot assessments by providing appropriate structuring to both the experience and recall aspects. By structure, it is implied that systematic exposure will be given to the relevant conditions, so that the pilot-rater has immediate experience with the work situations that are to be assessed. Also, the recall situation can be structured by use of standard inquiry procedures and aids to recall such as playback of recordings and provision of the part-task objective records.

In the B-737 flight tests, a combination of 14 FAA observers from both Washington, DC, and the Western Region, air carrier, air traffic control, and engineering, as well as observers from the Boeing Company, engineers from the 737 Control Cabin Equipment Group, and 737 Flight Test Pilots made up the technical evaluation team. Television and audio of all cockpit activities were provided in the passenger cabin area to supply all the observers with full information of cockpit action. The video, audio, and basic flight parameters were recorded to provide means for playback as well as on-the-spot evaluation. Every means had been taken to provide realistic inservice conditions so that actual workload could be observed under both normal and abnormal conditions. In addition, the video and audio portions of the flight were brought back and analyzed. As each flight was viewed, a team of people recorded with stop watches the number of minutes the crew could be observed or heard to be taking action in each of six categories:

Navigation - Consulting maps, and charts and setting RMI, course, flight director, and autopilot.

Communication - Sending or receiving communications.

Radio Tuning - Tuning radios for either navigation or communication purposes.

Controls - Actuation of flaps, gear, throttles, etc.

Systems - Operation of fuel, hydraulic, electrical pressurization, anti-ice, etc., during flight.

Miscellaneous - Includes all other observed cockpit actions needed in flying an aircraft such as checklist reading, writing down clearances, etc.

The resulting figures showed the time spent in climb, cruise, descent, and approach that the crew members used for each of the above categories. The remaining time was available for outside visibility and handling malfunctions. All tests have their limitations and this is no exception. Two of the major things that were not shown in this crew action study results are: (1) control movements that were so small as not to be detectable; e.g., small corrections in control wheel steering on the 737; and (2) the multiple tasks such as communicating while still maintaining outside observations. Of course, the previous disclaimer covering covert activities also applies; thinking, problem solving, planning, and feeling could not be assessed from the records.

At the end of a day of flight devoted to workload evaluation, it is usual to ask each crew member and designated observer to complete a standard form covering special features of the day's flight, such as weather and turbulence encountered on each leg of the trip and the type of approach conducted at each terminal. In addition, there is usually a more detailed record form that tallies specific problems such as completeness and correctness of all checklists, notation of necessary checklist interruptions, determination of the acceptability of crew workload upon special events such as a last-minute change in runway assignment, special air traffic routing, specific equipment outages, crew member incapacitation, and the like. These record forms are intended to assist the raters in covering all critical aspects of the flight and, thereby, avoid any "halo" effect as might occur when an overall successful operation causes one to forget some detailed aspect of the flight that was not satisfactory.

When these tabulations are completed on the day of flight, an analysis of the objective data will not yet be available. Prior to final acceptance by the FAA of the summed evaluations by pilots and observers, at least some sampling of the key objective records will have been made to ensure that tabulations made from event records provide supporting evidence for the pilot and observer evaluations. Each participating pilot and observer is asked also to state his rating of overall flight workload due to air traffic control procedures, communications, navigation, and collision avoidance and to rate the workload level for all procedures combined.

Very recently, a three-attribute rating of pilot mental workload has been developed to assist the assessor to explain observed workload in terms of the fraction of time busy, the intensity of required mental effort, and the intensity of emotional stress (reference 6). In completed certification programs, neither this three-attribute rating scale nor any other standard rating instrument has been used. Instead, rating procedures have been developed ad hoc by the flight test team and TCB.

The overriding purpose of all the airworthiness regulations is to ensure that the aircraft is safe. Crew complement determinations under FAR 25.1523 and workload evaluations made in connection with that regulation are then "assessments" that the flight deck design, both equipments and proposed/required procedures, are safe.

The widespread publicity given to the issue of crew size in the general press has led many outside the industry to think that workload assessment are merely quantitative; e.g., if the design presents excessive task demands, another crew member is required. This is not the case at all. The qualitative aspects of workload are clearly indicated in appendix D (FAR 25) as requiring evaluation, and it is the case that the nature of task demands and their compatibility with human abilities and limitations is very much a part of determination of regulatory compliance; (i.e., safety). Simply to emphasize this point, and not to imply that any such design feature has ever been found in certification testing, it is noted that overall workload could be low and still be unsafe. This could occur if: (a) the pilot's tasks were so automated as to become soporific; and (b) if some safety related procedure were so designed as to be easy to perform incorrectly or not at all. An example might be an essential control that violated human engineering guidelines (as specified in relevant documentation) and was incompatible in either operating sequence or logic of operation from other similar controls.

Controls and displays in today's airplanes are the result of the combined effort on the part of airframe manufacturers working with commercial airlines, instrument

manufacturers working with airframe manufacturers, the military services, and airlines and pilot groups working with engineering societies and industry committees. Of all this developmental interaction, the United States Air Force (USAF) has taken a leading position and has conducted cockpit development studies of more than 20 years (reference 33 and many additional reports). Due to this massive and continuous developmental effort on the part of all concerned parties, today's flight decks represent a great improvement in both quantitative and qualitative aspects of crew workload.

6.6.4 Remaining Problems.

Despite the improvements achieved to date, there is still room for improvement (reference 27). System complexity has increased in some domains, although partly compensated for by the great simplification in operation of turbojet engines and the thrust advantages of modern aircraft. The complexity factor alone justifies continued effort toward the best application of human engineering principles. A primary example is the means employed for interaction between the pilot and the avionic system. Commonly, this is a nonstandardized digit entry device and computer mode selector. At present, installed keyboards differ in logic of layout, labeling features, and sequence of mode selections. Several varieties of such digit entry devices are illustrated in figure 6-5. In addition to those shown, twin horizontal row arrangements of 1 to 5 and 6 to 0 are used in some avionics equipment found in existing aircraft certificated under FAR part 25.

The term "fractionation" was used in the early integrated cockpit program sponsored by the Armed Forces a generation ago (reference 33). The point made was that the pilot's efficiency may be compromised if he is forced to use equipment such as these various keyboards engineered by different people and showing various peculiarities. When the pilot must obtain disparate items of information from several different displays or make control movements and data inputs on several different devices, his speed of operation and assurance of correct performance are reduced.

The existence of "fractionation" can be determined by an analysis of what data the pilot must process in order to conduct required procedures. Such an analysis should be a part of the workload assessment procedure to provide assurance that design pitfalls in the area of fractionation, and particularly avionics input devices, are avoided. The opposite of fractionation is occurring with digital integrated systems. These systems have the capability of providing organized combinations of information on a single display. It is recommended that a fractionation data analysis be performed for keyboard data entry systems and for other subsystems that may present parallel possibilities of nonstandardization.

6.6.5 Need to Expand the Variety of Procedures.


As described in the preceding section, the available and presently well-proven methods of evaluating crew workload may be listed as: (1) Systems Analysis Methods; (2) Analysis of Pilot Performance by Time-and-Motion Studies and Related Simulations; and (3) Structured Rating Scales often employed in flight test programs. All of these have been used in recent certification programs with emphasis on making comparisons between the operability of the new flight deck and that of earlier models of related aircraft (e.g., the comparison of the DC-9-80 to the DC-9-50) or to modern 3-crew cockpits. Use of these methods has been successful, although each procedure has limitations, and improvements have been made in the way tests and simulations are conducted.

3 LETTER DESIGNATORS			PAGE FWD	ABC 1	DEF 2	GHI 3
CHG VHF	CHG WPT	FIX	PAGE BACK	JKL 4	MNO 5	PQR 6
VHF	WPT DATA	NAV DATA		STU 7	VWX 8	YZ 9
DIVERT	ROUTE	CHG ROUTE	SPACE BAR	CLEAR	Ø	INSERT
START	EXECUTE				ALERT	

ASTRONAUTICS CONTROL DISPLAY UNIT

1	N 2	3	MAN	AUTO	EM
W 4	5	E 6		THROT	DISPL
7	S 8	9	FLT PLAN	IDX	DIR
-	Ø	+	IFF	AAI	RADAR
•	L/L	/			UHF COMM
L	CLR	R	✓ LIST	MAP/RTN	FLT STAT

COLLINS CONTROL/DISPLAY
CONTROLLER KEYBOARD

RNAV ON	NAV 1 DME	RNG MONTR	ENRTE 5 400 NM FT 1.25 200 NM FT	APPR	F/D VNAV
FREQ	CRS	7	8	9	W/P
RADL	ALT	4	5	6	ROUTE R'CALL
RNG	VERT 	1	2	3	ROUTE STORE
ELEV	TOTAL DST TIME	PRG	Ø	ENTR	ROUTE CANCEL

J.E.T. DAC-2000

				LTR	ENTR	
A SCAN	B MAN WPT	C MAN TUN	D RTE	E 1	F 2	G 3
H STA	I WIND	J POS	K GS TK	L 4	M 5	N 6
O DST TIME	P XTK Δ ALT	Q DST CRSE	R SAT TAS	S 7	T 8	U 9
V WPT POS	W ALT FPA	X OFF SET	Y MSG	Z Ø	/	←

GARRETT AIRNAV 200

FIGURE 6-5. VARIETIES OF DIGIT ENTRY DEVICES

Additional and improved methods are needed for future applications. This has been acknowledged in many studies and reports published by industry and government authorities (references 2, 10, 11, and 14). It is not suggested that the already established methods of evaluating crew workload will be dropped or superseded; rather, it is expected that existing procedures will be supplemented by more specific tests which have greater validity/sensitivity and, as such, will be better predictors of crew/workload behavior.

Existing procedures for workload evaluation have never been fully satisfactory with respect to cognitive/mental activity as opposed to observable physical activity workload. The cost of extended flight test and the fact that design is largely frozen in production before completion of flight test make the one fully satisfactory method unavailable at the time the FAA must plan a certification program. The result is that flight test results yield little more than a binary decision — satisfactory (safe) or unsatisfactory (unsafe) — thereby contributing little to design improvement or reduction of design weaknesses that may be identified during the testing process. Since the manufacturer's own tests will have shown that the new aircraft compares favorably before it will be turned over to the TCB for flight test, it is not at all remarkable that all past flight tests have tended to result in favorable decisions (but in only nominal steps toward optimization).

The principal questions that need to be addressed by new workload assessment techniques are the following:

1. Do mental (cognitive/perceptive) task overloads occur in environments requiring complex operating procedures, under conditions of system failures (critical/crucial/noncritical) and during the switch to manual reversion conditions or when actually using manual back-up modes?
2. Do mental and activity underloads occur so as to tend to derogate vigilance?
3. Are there qualitative aspects to crew workload in new and derivative aircraft that may contribute to error potential and/or to unnecessarily increased cognitive processing demands?

The first question is addressed by current procedures, particularly task analysis and subjective ratings. The limitations of these methods make the presently available data incomplete, however, in that task analysis is dependent upon current assumptions and available data while pilot ratings may be subject to bias of various sources and be influenced by factors of pilot selection and the test situation.

To perform a totally satisfactory task analysis, it would be necessary to predict all the possible classes of workload events that the new flight deck systems are capable of presenting to the pilot — all the possible variable or insidious errors and fault combinations requiring pilot action (reference 34). Such an exhaustive listing is seldom available prior to long service experience. Similarly, while pilot ratings are easily obtained, it is difficult to ensure that the pilots included are fully representative of the future operators and that their reports of experience are valid replications of remembered earlier events. Data collected at the time of immediate experience would be more conclusive.

The second question, relating to mental underloads, also is capable of answer through current methods, but with limits. While accident investigations have often resulted in citation of behavioral phenomena thought to be capable of being caused

by underload (lack of vigilance, inattention to instruments, etc.), clear data do not exist showing the probability of such failures to follow practiced procedures as a function of specified (perceptual set, checklist behavior, nobody minding the store, etc.) mental task levels. This means that, at present, quantitative and empirical facts that would enable a judgment of safety to be made in relation to suspected mental underloads are not available. Regulatory decisions generally require a factual basis that is largely lacking using present methods on the question of mental underload.

The final question as to the quality of crew workload (and again, it is mainly the mental workload that is of concern) is also incompletely handled by present workload assessments. Standardization of controls and display and adherence to human engineering compatible with good engineering practice are points of attention in task analysis and other present tests. Going beyond those conventional checks, there is a present absence of proven methods for evaluation of the "psychological satisfactions" and rewards of pilot work. Challenge and mastery satisfaction are, of course, present in test situations, when the human operator knows his performance is being observed by his peers and evaluated. Novel work situations also tend to be stimulating. However, they can also cause an unwarranted amount of time to be consumed to the detriment of primary tasks.

Concern about the quality aspects of pilot workload relates to the situation of the average line pilot, not so much the pilot employed in developmental or certification testing of a new aircraft. The psychology of the line pilot differs from that of the test pilot in several ways. First, the novelty wears off in routine service over familiar routes, and second, well-practiced skills become less challenging with the result that level of effort to remain fully active in a largely automated task may decline. Furthermore, the nature of the crew scheduling procedure will sometimes place individual pilots together in a crew even though there may be personal incompatibilities or differences in preferred style of flight deck management, (i.e., anything from working with a chain smoker to assisting a rank conscious captain can create interpersonal strain further distracting the pilot from the mental energy expenditure that is required to remain in the data loop).

Increasing flight automation may result in reduced motivation for the average line pilot. Many studies have shown the preference of pilots for manual control with this preference being attributed to the fact that hands-on control supports the keeping of the pilot in the control loop (reference 35). Recognition of the importance of this point led the early pilot factors studies to investigate means ". . . to integrate the human pilot as an active element in an automatic flight control system" (reference 36). For example, in the above referenced report, the implementation of force wheel steering was studied as a method for permitting the pilot to make control inputs to the aircraft without disengaging the automatic flight control system. This system was implemented in such a manner as to allow the pilot to participate actively in the automatic control loop in an "easy and natural fashion." Because of the above factors, it has become more important than ever before to include, in workload assessments, an evaluation of the incentive and performance supportive aspects of workload demands that have been labeled the "qualitative" dimension.

6.6.6 Classification of Workload Assessment Methods.

Various writers have grouped present and newer workload measures in different classificatory schema. One format is to index methods by the type data recorded as

follows: (a) Behavioral measures such as primary task measures, secondary task measures, spare mental capacity, and subjective opinion; (b) Physiological measures such as electrocardiogram (ECG) or heart rate, electric brain recording (EEG), electrical skin conductance (GSR), electric muscle potentials (EMG), eye activity measured by oculometer/pupilometer, body fluids analysis for stress products, and respiration analysis; and (c) System analysis procedures such as computer models of part tasks (external search, time to complete various mental actions), baseline data studies (display metrics, cockpit noise level), and the time-and-motion studies that are often included in the cadre of pilot performance measures.

Secondary task methodology is sometimes treated as a separate category, and the term behavioral measures is then limited to measurement of aspects of the primary (flying) task. This is a convenient subdivision since a variety of secondary task methods exist, and several alternate versions of these procedures are finding current applications in pilot studies.

An alternative way of classifying measures of pilot mental and physical workload was specified by the Wright-State University project (reference 6.8.12). In the 1981 workshop held in Dayton, Ohio, Wright-State University researchers asked government, industry, and academic specialists to summarize individual measures falling in the areas of: (a) perceptual events; (b) mediational (thinking); (c) communications; and (d) motor activity. A physiological measure, for example, might be recommended as an index of any one or all of these pilot activities. Dividing workload techniques by this method has the advantage of pointing attention toward the pilot mental processes that may be completed correctly or that may result in error, and also this plan is congruent with the previous discussion of information flow (figure 6-2).

Since the immediate purpose of this chapter is to identify and select, from the broad area of emerging workload technology, candidate measures that may fruitfully be applied in near-term transport aircraft certification programs, it is proposed to follow the Wright-State classifications. These are preferred because the selection of a test method can be made in the light of the particular issues presented by the flight deck design in question, rather than by measurement modality, per se. By this it is meant that physiological procedures are to be adopted when and if they are needed because some particular question on pilot workload, relating to the various cognitive processes or motor activity, can best be answered through that form of measurement.

6.6.6.1 Measures Applicable to Perceptual Issues. Within the cockpit, stimulus events (e.g., warning tone, changed display indication, voice message), give rise to perception and increase mental awareness/workload. Thus, it may be important to have available a measure that shows when the pilot reacts mentally (perceives) the signal. Additionally, it may be desired to measure the magnitude of internal events, the emotional stress aroused by the processes initiated by the stimulus event, or the degree of mental concentration required to determine the exact nature of the problem and to resolve the necessary issues/problems at hand.

Two general measurement techniques are available which may indicate the time of perception: (1) electrical recordings from the brain or eye muscles and (2) eye movement-pupil reflex measures. Using the EEG or EMG techniques, electrodes placed on the scalp or near the extraocular muscles can be used to record potentials indicative of brain activity or eye movements. Similarly, direct observation of eye movements and/or pupil size changes may provide a record of when the eyes

were turned toward the stimulus event in question if of a visual character and the degree of concentration or stress associated with the visual perception (reference 37).

To measure the overall perceptual demands of flight activities, two other approaches have been developed. First, systems analysis methods provide for calculation of the perceptual demand complexity and demand for visual and auditory attention shifts that result from a particular layout and arrangement of displays. Second, computer models and simulation test procedures have been utilized to estimate perceptual activities in target search or external scanning. Both of these techniques have been used in the design stages of cockpit layout/workload assessment and phases of recent transport aircraft certification programs.

6.6.6.2 Measures Applicable to Mediational Workload. The EEG technique was particularly applied to pilot workload to permit estimation of ongoing mental activity such as thinking or problem solving. For example, an unobtrusive visual signal (e.g., a 60-cycle flicker) generates EEG recorded signals. Particular portions of the record can be computer analyzed to show amplitude and latency changes that correlate with the ongoing intensity of concurrent mediational processes. Similarly, electrodermal probes have been used in the laboratory. It has been shown that measures such as galvanic skin resistance (GSR) risetime, recovery rate, and amplitude correlate to some degree with the intensity of cognitive activities. The sequence of events may be so intense that thinking or problem solving activity leads to a degree of emotional stress producing palmar sweating which can be recorded and correlated with other physiological activities and performance measures. Finally, cumulative mental workload has been estimated by measures of body fluids. Chemical by-products resulting from mental stress may be isolated in these samples.

6.6.6.3 Measures Applicable to Communication Workload. Time-and-motion studies of pilot performance, such as the time-line evaluation model used by Boeing Commercial Aircraft Company or other somewhat comparable alternative procedures, provide estimates of the communication workload. A specific verbal activity channel workloading may be computed as a function of alternate procedures and overall task demands. Computer models based on systems analysis methods and observations of actual pilot performances have been developed by DOD as well, particularly the Air Force at Wright-Patterson Air Force Base and the Naval Air Development Center. Communications activity is one aspect of the estimated total workload from these models.

One of the simplest methods proposed as a measurement technique is use of a decibel meter. The cockpit noise/communication ambient is measured directly and recorded on a strip chart. The assumption made is that communication workload is directly related to cockpit noise volume in some types of situations and is inversely related to noise in some others. Aural warnings, of course, contribute to the measurements.

Secondary task procedures have been used to measure communications workload. The assumption is made that as the timely and precise accomplishment of required verbal communications increases, "busyness" prevents completion of more of the assigned secondary task. Because procedures of this kind may be used to estimate nearly any form of "busyness," not only that due to communications demand, secondary task methods will be discussed in a separate section below.

When used specifically for communications workload measurement, the procedure begins by scaling for difficulty the variety of required pilot communication messages. The communication tasks are then performed while the pilot is engaged in the primary flight control procedures. Primary task performance may be evaluated by records of control reversals, clearance deviations in altitude, heading or speed, etc. The assumption is that optimum communications performance and optimum primary task performance will be incompatible at some high level of workload.

Finally, voice stress analysis has been used. As a measure of the stress resulting from workload when the pilot is required to conduct voice communications and fly complex procedures, a voice recording is computer analyzed for tremor, pitch, and articulation changes.

6.6.6.4 Measures Applicable to Motor Workload. Total motor workload in a mission or mission phase can be estimated by EMG spectra related to muscle fatigue. Muscle tremor as well as electromyography has been recorded for pilot operations considered to be particularly fatiguing, e.g., helicopter flying.

In the shorter time span, a variety of measures of circulation have been used. Beat-to-beat variability in heart muscle action has shown direct relation to motor workload as well as stress workload resulting from task criticality. Similarly, raw heart rate, blood pressure, and blood pressure changes have been found in various situations to correlate with motor work, time on task, and stress and fatigue. Studies have not been universally positive, perhaps due to interactions of measures with respiration and other activities or changes. Ease of recording and calculation are general advantages.

Specific electrocardiography (EKG) devices have been constructed so as to be noninvasive, but still produce measures of stroke volume, ventricular ejection time, and other facets of heart action. Ultrasound is one method employed, but this method is reported to involve difficult instrumentation and skilled interpretation of data. Impedance cardiography is an alternate noninvasive method capable of obtaining data on central cardiac parameters (heart rate, cardiac output, stroke volume etc.) and may provide good correlations with data obtained by other means, particularly if each subject is used as his or her own control.

Respiration rate and variability in respiration can be monitored easily in operational environments and have shown relations to workload. Again, laboratory studies have not been universally positive in outcome, and this may suggest contamination of data by variables such as subject motivation.

Body fluid analysis is a diffuse technique that cannot be aimed directly at one short segment or another of total workload. Urine samples in particular have been used to indicate the presence of stress/fatigue by-products. Total workload may be estimated from total catecholamine excretion, and some studies suggest that physical versus mental work may be estimated by proportions of constituents such as epinephrine and norepinephrine. The technique makes timing of sample collections critical, and there is no easy way to eliminate contamination from nontask factors such as time of day, diet, and temperature stress.

Eyeblink analysis based on data obtained with electrodes placed near the eyes has been recommended as a measure of fatigue and has been shown to change with time-on-task and with various drugs. Both the duration of the blink and the recurrence pattern are examined by computer, and at the same time saccadic velocity may be recorded.

In addition to these newer procedures, motor workload has been measured for many years by computer modeling of dimensions and task demands, by strain gages placed on controls, by record of direct pilot performance in tracking altitude, heading, and speed, and by control reversal markers.

6.7 RECOMMENDED APPLICATIONS.

At the Mati, Greece, Symposium (reference 14), it was proposed that an attempt be made to produce a table of recommended workload assessment applications. A format for such recommendations was proposed with the types of pilot work on one axis: (a) producing force; (b) performing sensory-motor coordinations; (c) making decisions and choices of preferred actions; (d) monitoring, (e) scanning, and otherwise searching for information; (e) generating new or creative problem solutions; and (f) supervising and coordinating crew activities. The other suggested axis of the table provides a place for entering recommended methods of evaluating the magnitude and acceptability of workload requirements broken down as to the internal processes that occur in the mental work sequence: (a) Sensory-perceptual activities, (b) mediational (thinking) activities, (c) communications, (d) and motor control action.

6.7.1 Specific Procedures.

Figure 6-6 presents one attempt to block in such a table of recommended applications. No such table can ever be all-inclusive or fully congruent with the needs of the evaluator in all possible issues and unique test situations. It is possible, however, that a clearer perspective may be attained on the availability and possible applicability of the newer, and as yet less tested, workload assessment methods in the civil aircraft realm. For this reason, figure 6-6 is given as an illustration of a set of recommendations that is subject to review and further improvement.

The population of methods of evaluating pilot workload, each described in more detail in earlier sections, is given in figure 6-7.

Reference to the earlier section on current workload assessment, will show that of the total of 21 evaluation method categories presented above, only 11 have found extensive use in transport aircraft proving programs. Six of the 11 are the 6 given systems analysis methods, the others being subjective ratings by pilot and observer in flight test, and in mock-up or simulation situations, and behavioral measures of primary task performance, synthetic task performance--part task, and flight test. From a simple count, past programs have used just over one-half the available methods.

Checking the issue-application boxes in figure 6-6, it appears that several categories of recommended measures have not yet been used extensively in civil programs. These under-utilized procedures are the physiological quintet, full-mission simulation procedures, and a number of the behavioral measures including

METHODS OF EVALUATING WORKLOAD RE:

<u>TYPE OF WORK</u>	<u>SENSORY PRECEPTUAL</u>	<u>COGNITIVE MEDIATIONAL</u>	<u>DATA INPUT-OUTPUT COMMUNICATION</u>	<u>MOTOR CONTROL ACTION</u>	<u>TOTAL MISSION</u>
1. Producing Force			Simulation Part Task Systems Anal Modeling	Time + Motion Strain Gauges Physiological	1. Subj. Rating/Fit Test 2. Physiological 3. Primary Task/Fit Test
2. Sensory-motor Coordination	Simulation, Part Task Systems Anal Time and Motion			Simulation, Part Task Systems Anal Physiological	Same as Above
3. Decision Making	Systems Anal Simulation, Full Mission Subj. Rating	Systems Anal Simulation, Full Mission Physiological			Same as Above
4. Search for Information	2nd Task Sim. Simulation, Part Task Modeling	2nd Task Sim. Simulation, Part Task Subj. Rating			Same as Above
5. Creative Solution	Simulation, Full Mission Systems Anal Subj. Rating	Physiological EEG Systems Analyses Subj. Rating	Simulation, Full Mission Systems Analyses Subj. Rating		Omit Physiological Others same as Above
6. Supervision Crew Coordination	Simulation, Full Mission Subj. Rating, Observer Body Fluids Physiology	Simulation, Full Mission Subj. Rating, Body Fluids Physiology	Subj. Rating Simulation, Part Task DB Meter		Omit Physiological Others same as Above

FIGURE 6-6. INVENTORY OF AVAILABLE WORKLOAD METHODS AND ISSUE/APPLICATION SITUATIONS

1. Subjective Rating, obtained in:
 - a. Flight test, self
Flight test, observer
 - b. Full mission simulation, self
Full mission simulation, observer
 - c. Mockup, simulation
2. Physiological measures, class of:
 - a. EKG, respiration, etc.
 - b. EEG, GSR, etc.
 - c. EMG
 - d. Body fluids analysis
 - e. Eye movements, pupil diameter, blink rate
3. Systems Analysis, measurement and comparison by:
 - a. Computer modeling
 - b. Time-and-motion study
 - c. Simulation of part tasks
 - d. Checklist, procedures comparison
 - e. Standard operating procedures, training requirements
 - f. MEL comparison
4. Behavioral Measures of:
 - a. Primary task performance
 - b. Secondary task performance
 - c. Voice stress analysis
 - d. DB meter
 - e. Synthetic task performance, part task
 - f. Full mission simulation
 - g. Flight test, mini-airline

FIGURE 6-7. POPULATION OF METHODS OF EVALUATING PILOT WORKLOADS

secondary task performance, voice stress analysis, and decible (db) meter. This enumeration does not mean that each possible method is equal in potential yield of new information.

As has been noted before, flight test supported by appropriate systems analyses, and synthetic task studies have provided the answers needed to all crucial past workload questions. The interest in use of a broader sampling of available methods arises from the potential time and cost advantages (over costly and late flight test results) and the potential ability to reveal qualitative differences and measures of covert, hard to observe aspects of mental workload. Stress arising from the forced choice between acceptance of automated but incomprehensible problem solutions or manual reversion to little practiced skills, for example, might be revealed more usefully by physiological measures than by the subjective ratings now relied on. This has been described as stress arising from "lack of trust in the displayed information" (reference 14), and it seems likely that such stress will be a more important workload component with increased automation.

6.7.2 Combined Methods.

From the discussion of individual workload evaluation methods, it should be clear that a high degree of overlap or commonality exists. The same method may be useful for measurement of various types of work, and there are several methods recommended for most applications. This is similar to the problem of assessment in any complex and dynamic environment — no one sign or threshold quantity can be used to describe fully the situation. Rather than making a long series of measurements using first one technique and then another, it is recommended that the workload be evaluated simultaneously by multiple methods during most tests. The tests themselves are costly and time consuming. The maximum data output should be sought in each simulation or test flight.

Physiological methods are a prime example of the potential use of combined methods. There are known interactions among different measures of body functioning that make analysis of data easier and more profitable if control data on several functions are collected at once. Hence, a package of physiological measures is more powerful as an analytic tool than any group of individual tests done in sequence.

As in many areas of workload technology, the military services have sponsored development of advanced technology in physiological monitoring of pilot workload. Various recording packages have been developed and tested to provide a record of pilot circulation, respiration, muscular and skin action, and brain activity. These may be highly useful as combined measures, if, for example, it is desired to know when a warning signal is received (perception), how it is processed (mediation), and when it is acted upon (motor). EMG records can show when eye movement occurred, EEG can show an evoked potential representing pilot recognition of an important event, heart action, respiration and skin sweating may show stress accompanying the search for a decision, and muscle potentials may show the moment and character of control actions. Such a physiological record, supplemented by primary task records of flying events would allow a comparison, in depth, between two alternate designs or procedures. The reaction time/latency data combined with magnitude of activation indications, if paralleled by performance measures, would have more probative value than any one physiological measure taken alone. For this reason, it is recommended that a civil aviation pilot monitor device be utilized. The design goal should be to obtain repeatable (reliable)

measures that do not interfere with normal pilot freedom of motion, that are readily scored, and have pilot acceptance. Cause-effect relationships must be demonstratable for the use intended (validity).

Combined computer models have been developed by the Air Force and the Navy to assist in both the design of pilot environments and their evaluation. For the design phase, major aircraft manufacturers have also made extensive use of individual and combined method computer models. For example, a model that generates data on internal visual requirements (how often, how far, and for how long the eyes must be deflected to monitor instruments and indicators), has been combined with a model of external vision (cockpit cutoffs, areas of search for high probability traffic, time required for detection) to give an estimate of overall visual channel loading. The visual channel model may then be combined with other models to estimate total task demands. Motor workload in reaching and moving controls, inserting keyboard and other data, and operating communications equipment may be combined with the visual model, and further with other models of mental workload (decision requirements, planning, speaking, etc.).

The workload assessment model (WAM) designated as CAFES calculates workload in each channel (visual, mediation, verbal-auditory, motor) for each flight phase. Following preparation of a mission scenario with specification of the tasks required for each phase, a time-line is prepared. A statistical computation generates workload estimates as a function of required task-time versus time available for each pilot channel (reference 38).

The Naval Air Development Center has developed a model to simulate the human operator in complex systems. There are three major related programs: Human Operator Simulator (HOS), HOPROC Assembly Loader (HAL), and Human Operator Data Analyzer and Collator (HODAC). The HOS program accepts inputs from HAL, which includes the operator's long-term memory, and inputs from the user on the design of the work station. HODAC is a series of subroutines engaged in analysis of the HOS generated data. HOS operates like a real human not simply an unthinking robot. It is goal-oriented and adaptive, making use of a series of micromodels for various human behaviors. While development of such an overall model is unlikely ever to totally duplicate all human variations, it is reported that many human performance predictions generated by HOS confirm earlier empirical studies of actual performance. Known effective displays are found to be better used, etc. (reference 39).

Crew performance has been evaluated by the Air Force, drawing on systems analysis work by McDonnell-Douglas and the Boeing time-line method, combining measurements made in operational tests. Along with task-time measurements made during pre-flight, flight, and post-flight phases, evaluations were obtained from safety observers who were on the aircraft and questionnaire data were provided from crew observation and debriefing. From all that data, and the earlier task analysis, relatively valid task-load measurements were derived (reference 40). From the examples given above, it is clear that overall predictions of crew performance made before flight test and also developed with the addition of data from a brief flight test series can be improved by using a combination of the most advanced methods of analysis.

6.8 REFERENCES

1. FAR Part 25, "Airworthiness Standards: Transport Category Airplanes," Department of Transportation, Federal Aviation Administration, June 1974.
2. Chiles, W. D., "Objective Methods for Developing Indices of Pilot Workload," FAA Report No. AM-77-15, July 1977.
3. FAA Order 8110.4, "Type Certification," Department of Transportation, Federal Aviation Administration, December 1967 (Revised 1978).
4. Rolfe, J. and Lindsay, S. J., "Biological Measures of Workload," Royal Aeronautical Society, Proceedings of the Symposium on Flight Deck Environment and Pilot Workload, March 1973.
5. Proceedings of the NATO Science Committee Symposium on Mental Workload Measurement," Mati, Greece, September 1977.
6. Sheridan, T. B. and Simpson, R. W., "Toward the Definition and Measurement of the Mental Workload of Transport Pilots," Massachusetts Institute of Technology, FTL Report No. R 79-4, January 1979.
7. Chiles, W. D. and Alluisi, E. A., "On the Specification of Operator or Occupational Workload with Performance-Measurement Method," Human Factors, 21, 1979, pp. 515-528.
8. Cohen, S. I. and Silverman, A. J., "Measurement of Pilot Mental Effort," Flight Test Techniques Panel Report No. 148, London, England, May 1957.
9. Gomer, F. E., "The Application of Biocybernetic Techniques to Enhance Pilot Performance during Tactical Missions," McDonnell-Douglas Report No. MDC-E 2046, October 1979.
10. Reising, J. M., "The Definition and Measurement of Pilot Workload," USAF Report No. AFFDL-TM-72-4-FGR, Wright-Patterson Air Force Base, Ohio, February 1972.
11. Kitay, D. S., "Proceedings of the Symposium on Man-System Interface: Advances in Workload Study," Airline Pilots Association, Washington, D. C., August 1978.
12. Mohler, S. R., Sulzer, R. L., and Cox, W. J., "Workshop on Crew Workload Measurement," (Report of a Conference held at Wright-State University in February 1981), in press.
13. Mohler, S. R., Personal Report of Consultant Visit to AIRBUS Industries, 1981.
14. Parks, D. L., et al, "Mental Workload in Systems Applications — Applications Workshop Summary," NATO Symposium on Mental Workload, Mati, Greece, September 1977.

15. Peal, R. A., "The 767's Flight Management System — A New Generation of Airborne Avionics," *Astronautics and Aeronautics*, 18 November 1980, pp. 37-39.
16. McLucas, J. L., Drinkwater, F. J., and Leaf, H. W., "Report of the President's Task Force on Aircraft Crew Complement," July 2, 1981.
17. Stone, G., Regis, E. R., and Gulik, R. K., "Executive Summary — DC-9 Super 80/DC-9-50 Comparative Flight Crew Workload Study," Douglas Aircraft Company, Report No. MDC-J 8749, August 1980.
18. Boucek, G. P., et al, "Aircraft Alerting Systems Standardization Study — Vol. 1 and 2," Federal Aviation Administration, Department of Transportation, FAA Report No. FAA-RD-81-38, 1 and 2, January 1981.
19. Krebs, M. J., Sandvig, J. H., and Wolf, J. D., "Color Display Design Guide," U.S. Navy, Office of Naval Research, October 1978.
20. McFarland, R. A., "Human Factors in Air Transportation," McGraw-Hill, New York, 1953.
21. "The Automatic Complacency," The Cockpit, United Airlines Corporation, Flight Operations Newsletter, April 1978, pp. 2.
22. "Special Air Safety Advisory Group Report to the Federal Aviation Administration," Department of Transportation, Federal Aviation Administration, FAA Report No. AFS-1-76-1, July 1975.
23. Eldredge, D., Crook, W. G., and Crimbring, W. R., "Simulation Tests of Flight Technical Error in 2D/3D Area Navigation (RNAV), using a Multiple Waypoint RNAV System with and without a Flight Director System," Department of Transportation, Federal Aviation Administration, FAA Report No. RD-77-112, October 1977.
24. Boehm-Davis, D. A., et al, "Human Factors of Flight-Deck Automation — NASA/Industry Workshop," NASA Report No. TM-81260, National Aeronautics and Space Administration, 1981.
25. Jordan, N., "The Allocation of Functions between Man and Machines in Automated Systems," *J. Appl. Psychol.*, 17, pp. 161-165.
26. Wiener, E. L., "Controlled Flight into Terrain Accidents: System Induced Errors," *Human Factors*, 19, 1977, pp. 171-181.
27. Hay, G. C., House, C. D., and Sulzer, R. L., "Summary Report of 1977-1978 Task Force on Crew Workload," Department of Transportation, Federal Aviation Administration, FAA Report No. EM-78-15, December 1978.
28. Sulzer, R. L., "Transport Aircraft Cockpit Standardization," FAR Part 25, Department of Transportation, Federal Aviation Administration, FAA Report No. EM-81-11, November 1981.

29. "Delta Airlines DC-9-71, Boston, Massachusetts, July 31, 1973," National Transportation Safety Board, Report No. NTSB-AAR-74-3, March 1974.
30. H. P., Ruffell-Smith, "A Simulation Study of the Interaction of Pilot Workload With Errors, Vigilance and Decisions," NASA Technical Memorandum 78482, NASA AMES Research Center, Moffett Field, California, June 1979.
31. Sulzer, R. L., Cox, W. J., and Mohler, S. R., "Flight Crewmember Workload Evaluation," Department of Transportation, Federal Aviation Administration, FAA Report No. RD-80-129, ASF-80-5, April 1981.
32. FAR Part 121, "Air Carriers and Certification and Operations: Domestic, Flag, and Supplemental Air Carriers and Commercial Operation of Large Aircraft," Department of Transportation, Federal Aviation Administration, April 1974.
33. "Whole Panel Control Display Study," USAF Report No. ASDR-61-91, July 1960.
34. Bird, G. T., and Herd, G. R., "Experienced In-Flight Avionics Malfunctions," in AGARD Lecture Series Report No. 81, "Avionics Design for Reliability," AGARD Report No. LS-81, March 1976, pp. 4.1-4.10.
35. Murphy, J. V., et al, "Integrated Cockpit Research Program: Vol. I, Final Report," U.S. Navy, Office of Naval Research Contract NONR 4951 (00)-NR 213-041, January 1967.
36. Swartz, W. F., "Control-Display Pilot-Factors Program," USAF, IPIS, Randolph Air Force Base, Texas, September 1966.
37. Young, L. R., and Sheena, D., "Survey of Eye Movements Recording Methods," Behav. Rsch. Metho. and Instrumentation, 7, 1975, pp. 397-429.
38. Strick, M. I., "The Human Operator Simulator," Analytics, Willow Grove, Pennsylvania, 1978.
39. Wherry, R. J., "The Human Operator Simulator in the Identification of Potential Problem during System Development," in Kitay, D. S. (ed), and "Proceedings of the Symposium on Man-System Interface: Advances in Workload Study," Airline Pilots Association, Washington, D. C., August 1978.
40. Geiselhart, R., "Crew Factors, Workload and Performance," in Kitay, D. S. (ed), "Proceedings of the Symposium on Man-System Interface: Advances in Workload Study," Air Line Pilots Association, Washington, D. C., August 1978.

6. APPENDIX

PROPOSED PROCEDURE FOR MINIMUM CREW DETERMINATION

HANDBOOK GUIDELINES.

The Federal Aviation Administration (FAA) publishes orders intended to standardize the application of regulations in the form of individual handbooks, each devoted to an area of rules. The two such orders most relevant to minimum crew determination for transport category aircraft are 8110.4, "Type Certification," (reference A-1) and reprinted with summary changes 1 through 22 in October 1978, and 8110.8, "Engineering Flight Test Guide for Transport Category Airplanes," (reference A-2). Other relevant guidance information is found in various advisory circulars and in standards and recommended practice documents prepared by Radio Technical Commission for Aeronautics (RTCA) and Society of Automotive Engineers (SAE).

Within any given FAA handbook, major subjects are grouped in chapters, and individual regulatory topics are treated in numbered paragraphs, some of which are reserved for future preparation of additional guidance information. In Order 8110.4, there is no specific paragraph devoted to minimum crew determination. In the case of Order 8110.4, however, a paragraph proposal, Number 187, "FAR 25.1523, Minimum Flight Crew," has been drafted and discussed within the FAA but has not been approved in final form to set uniform procedures in place of the present reserved Paragraph 187.

The purpose of this section is to present the author's recommendations for explicit statement of the procedures to be followed by Engineering and Manufacturing personnel, Washington and field, in accomplishing the implementation of FAR 25.1523 and appendix D. The present draft material draws on earlier drafts prepared by FAA Regional Certification Engineers but not approved by FAA headquarters. It is recognized that current implementation of the philosophy of Regulation By Objective (RBO) and the relatively new assignment of Lead Region responsibility to the Northwest-Mountain Region (ANM), in Seattle, may require substantial modification of procedures developed earlier and approved through different FAA channels. Hence, this material is intended to be suggestive of general content but should not be viewed as in any way authoritative or final.

GENERAL POLICY.

Recognizing that flight crew workload is affected by Air Traffic Control (ATC) and airway procedures and facilities, and that the ATC system is being changed to improve safety, efficiency, and controller productivity by increased automation and related enhancement activities, it is necessary to state a general policy or principle, to wit:

All improvements to ATC and airway procedures and facilities will be tested in advance of implementation to ensure that they do not adversely impact flight crew workload.

Observance of this principle is necessary because airworthiness certification is conducted in the context of a particular level of ATC service and related crew

workload. An aircraft approved for operation by a stated minimum crew, and configured to provide appropriate working stations for only that stated crew size, will have a long service life, covering the period of many changes in ATC and airway procedures and facilities. An obvious way to reduce the Federal cost of ATC services would be to transfer some of the controller workload to aircrews; (e.g., by handling increases in traffic density with airborne systems rather than ground separation assurance.), however, the flight deck crews, through their union association, have begun to express concern that pilot workload has been increased by ATC changes. Fortunately, studies to date have not shown any related decrease in safety, but this past record alone is not sufficient to guarantee that all future system changes will be equally well accommodated. Hence, adherence to a firm policy of pre-test for crew workload adverse impact is necessary.

DRAFT PARAGRAPH "187 FAR 25.1523 MINIMUM FLIGHT CREW" FOR ORDER 8110.8.

a. Explanation

(1) Under this rule, the minimum flight crew for a transport aircraft must be established so that it is sufficient for safe operation considering:

(a) The workload on individual crew members.

(b) The accessibility and ease of operation of necessary controls by the appropriate crew members.

(c) The kind of operation authorized under FAR 25.1525. The criteria used in making the determinations required by this section are set forth in appendix D of FAR 25.

(2) The procedures for determining compliance with FAR 25.1523 and appendix D may vary in depth depending on whether the certification is:

(a) A new model.

(b) A follow-on model.

(c) A modification made to an already approved aircraft for the specific purpose of changing the number of crew members and/or crew duties.

(3) Preliminary discussions with the manufacturer appropriate to crew workload should be made early in the developmental cycle. This discussion should lead to agreement as to the identity of new instrument/system displays or revised groupings of such indicators, or advanced levels of automation that require particular evaluation to determine the validity of either assumed increases in pilot performance or reduced workload. Simple reduction in the quantity of crew workload is not an acceptable design goal without accompanying demonstration of crew duties and work requirements for each crew member that preserve an important and active role (qualitative aspect of workload).

The initial design decisions made at a stage of the aircraft program when an actual production aircraft or a full-performance ground flight simulator is not available are important in providing data on alternate design features that will not be later demonstrated in the prototype flight deck. Because these design stage studies are the main source of data showing that best choices have been made for the prototype, it is essential that those pre-certification tests be selected from the most

appropriate and up-to-date workload measurement methods and that those early tests be conducted with rigor and FAA oversight.

Final FAA decisions for minimum crew determination cannot be made until the aircraft is test flown and evaluated throughout the FAA flight test certification program and in a simulated airline operation.

(4) Examples of aircraft for which a full program of FAR 25.1523 compliance demonstration (a "full program" is one that is independent of any earlier test programs and includes new demonstrations of suitability and acceptability of all aspects of crew activities as summarized in appendix D) are as follows:

(a) Any transport aircraft operated under Part 121 designed to be flown by a minimum crew of two pilots, or

(b) A design incorporating major increases in automation, and making the assumption of favorable impacts on pilot workload.

(c) A changed design providing for removal of conventional raw data displays and important crew-adjustable controls if such design presents potentially difficult trouble-shooting or failure detection to the flight crew, major change in the flight control system, or in basic flying qualities.

When a full program is required, there will be FAA participation in planning and conduct of design stage evaluation tests made before the final specification of flight deck configuration. All data collected during those design stage tests will be summarized, interpreted, and submitted to FAA as part of the application for certification. When the flight test portion of a full program is planned, arrangements will be made to ensure that subject pilots are representative of airline crews and include a full range of ability.

b. Procedures

(1) A systematic evaluation and test plan is required for any new aircraft. The methods for showing compliance should emphasize the use of the latest analytical, simulation, and flight test techniques. The crew complement should be studied through a logical process of estimating, measuring, and then demonstrating the workload imposed by a particular flight deck design.

The analytical measurements should be conducted by the manufacturer early in the aircraft design process. The analytical process which a given manufacturer uses may vary, since there is no single valid and accepted method for determining crew workload.

(2) Analytical Approach

One acceptable analytic approach defines workload as a percentage of the time available to perform the tasks. This is accomplished by dividing the time required by the time available x 100 to convert to a percentage. This process must be applied to all phases of flight such as taxi, takeoff, climb, cruise, descent, approach, and landing. This method is satisfactory for evaluation of flight deck changes relating to overt pilot work such as control movements and data outputs. Planning and decisionmaking impacts of changes and mental workload resulting from automation changes are less subject to ready quantification by this method.

Absolute standards are not available for interpretation of obtained time required scores, but those records can be used to identify high or simultaneous workload demands for later test in a simulator or aircraft, and comparisons can be made with overt workload demands in proven aircraft. Detailed procedures for doing this work have been published by E. L. Brown, G. Stone, and W. E. Pearce, "Improving Cockpits Through Flight Crew Workload Measurements" (reference A-3). Under these procedures, an ultimate summarization covering many possible flight situations can be made by a computer that has the capability to process large numbers of possibilities and combinations.

The most frequently used basis for decision that a new design is acceptable is a comparison of a new design with a previous design proven in operational service. By making specific evaluations using the latest human factors technology, and comparing new designs to a known baseline, it is possible to proceed in confidence that the changes incorporated in the new designs represent the significant workload improvements that were intended. When the new flight deck is considered, certain components may be proposed as replacements for conventional items, and some degree of rearrangement may be contemplated. New avionics systems may need to be fitted into existing panels, and newly automated systems may replace current indicators and controls. The general configuration of the flight deck is likely to remain quite familiar in the new model to the existing and proven design. As a result of this evolutionary characteristic of the flight deck design process, there is frequently a reference flight deck design by which is meant a conventional aircraft that has been through the test of airline usage. If the new design represents an evolution, improvement attempt, or other derivation from this reference flight deck, the potential exists to make direct comparisons. While the available workload measurement techniques do not provide the capacity to place precise numbers on all the relevant design features in reference to error or accident potential, these techniques do provide the required means to compare the new proposal to a known quantity, an accepted cockpit proven by operational experience to have satisfactory workload characteristics. However, service experience should be researched to assure that any existing problems are understood and not perpetuated.

While a subject pilot, after studying a new component or arrangement and exercising it in practice flight scenarios, may not be able to grade that design in finer workload units than those of a Cooper-Harper type of scale, the pilot can say with reliability and confidence that it is or is not easier to see a display or to use an augmented control system than to use a parallel unit of a reference design. These "better" or "worse than" judgments, if corroborated by a reasonable sample of pilots over various assumed flight regimes, provide substantial evidence that workload is or is not reduced by the innovation.

If a new design represents a "revolutionary" change in level of automation or pilot duties, analytic comparison to a reference design will have lessened value. Without a firm data base on the time required to accomplish both normally required and contingency duties, more complete and realistic simulation and flight test will be required.

(3) Aircraft Testing

In the case of minimum crew determination, the final decision is reserved until the aircraft has been flown by a panel of experienced pilots, particularly in the case of any new aircraft that is designed for operation by a smaller crew than the earlier or comparison aircraft. More assurance is derived from actual flight

tests than from earlier simulator tests or other synthetic or computer model procedures.

Appendix D of FAR 25 (reference A-4) contains the criteria for determining the minimum flight crew under FAR 25.1523. These criteria contain:

Basic workload functions.
Workload factors.

The workload factors are those factors which must be considered when evaluating the basic workload functions. It is important to keep in mind the key terms basic workload and minimum crew when analyzing and demonstrating workload. For example, an evaluation of communications workload should include only that basic workload required to properly operate the aircraft in the environment for which approval is sought. It is neither necessary nor appropriate to load a crew member with an inordinate amount of company-required radio communications. This basic philosophy is important to keep in mind if a consistent evaluation of minimum crew is to be accomplished.

The flight test program for showing compliance with appendix D of FAR 25 should be proposed by the applicant and should be structured to address the following factors:

Route: The routes should be constructed to simulate a typical area that is likely to provide some adverse weather and IFR conditions, as well as a representative mix of navigation aids and ATC services.

Weather: The aircraft should be test flown in a geographical area that is likely to provide some adverse weather such as turbulence and IFR conditions.

Crew Fatigue: The crew should be assigned to a daily working schedule representative of the type of operations intended. The program should include the duration of the working day, the number of departures and arrivals, and the number of successive days a crew would be expected to work.

Minimum Equipment Test: The MEL should be reviewed for inoperative items that would result in added workload. Critical items and reasonable combinations of inoperative items should be considered in dispatching the aircraft.

Traffic Density: The aircraft should be operated on routes that would adequately sample high density areas, but should also include both precision and nonprecision approaches.

Incapacitated Crew Member: The program should include a demonstration of the total incapacitation of a crew member at any point in a given flight. It must be shown that the aircraft can be safely operated to the destination with the remaining crew.

Systems Failure: Consequences of reversions from normal to failed modes of operation should be included in the program.

Emergency Procedures: A sampling of various emergencies should be established in the test program to show the effects on crew workload.

(4) Determining Compliance Appendix D

The type certification team that serves as pilots and observers should be equipped with flight cards or other forms that allow for recordkeeping of comments addressing the basic workload functions as delineated in the 10 workload factors. These records should be accumulated for each flight or series of flights in a given day. For the purposes of this data gathering, the aircraft should be configured to allow the team members to see all crew activities and hear all communications both externally and internally. Some previously conducted programs have used closed circuit TV cameras with "hot" mikes to accomplish this requirement.

Each paragraph of appendix D summarizes an observation of pilot performance that is to be made. Judgment by the certification team members must be that each of these tasks has been accomplished within reasonable workload standards during the test flights. Numerical pass-fail values cannot be attached to these listed observations in the sense of an absolute scale going beyond relative judgments such as those given in a Cooper-Harper type rating scale. Furthermore, a holistic pilot evaluation rationale is needed in view of the wide variety of possible designs and crew configurations that makes it unfeasible to assume that ratings are made against every alternative and against some optimum choices. The criteria, then, do not permit finely scaled measurements. Specific feature and activity pass-fail judgments should be made, the pass being that the aircraft meets the minimum requirements.

(5) Additional Workload Test Methods

If a new aircraft design presents major issues not subject to ready test by the more conventional test methods summarized above, appropriate test methods will be selected from the larger variety of existing simulation, pilot performance measurement, and pilot physiological techniques. For example, if the acceptability of mental workload demands is at issue, it may be necessary to conduct dynamic ground simulation in engineering or developmental cab type environments. Comparisons may be required between the speed and accuracy of problem resolution in a conventional and modified flight deck design or with conventional versus modified handling qualities.

To evaluate stress and fatigue effects of mental and physical workload, it may be necessary to record indices of physiological activity such as brain potentials, eye muscle or effector organ muscle potentials, skin conductivity, or heart action. Eye movement and pupil action may require study to compare new visual task loadings. In any case, it should not be assumed that all new designs can be evaluated exactly as was done in the past. The latest proven workload technology should be utilized.

(6) Coordination of the Workload Demonstration Plan

The responsibility for preparation of the data collection and analysis plan rests with the manufacturer, who will be supported and aided by FAA specialists who have knowledge of previous type certification activities. Among the responsibilities of the manufacturer, in preparing this plan, is to insure that the best available information from several outside sources is considered. For example, airline safety and training departments should be consulted to develop a list of actual failures and peak workload situations that have been known to occur in recent and representative airline experience. NASA, DOD, and FAA engineering and development agencies should also be consulted by the manufacturer in connection with his workload demonstration plan development. One goal of this outside

consultation would be to insure that the latest research data and improved methods of workload evaluation are available to the certification team. Another responsibility of the manufacturer at this final planning stage will be to summarize and present the views of pilot groups and recognized collective-bargaining organizations as to the potential pitfalls of workload testing with new aircraft. It is recognized that a practical evaluation plan cannot include all possible variations of method and cannot cover all possible operating environments. It is important, however, that the choices of what methods to utilize and what conditions to represent should be made after consultation with all competent and interested parties. Certification is an FAA responsibility with the support of the manufacturer, but other organizations should have every opportunity to propose and recommend what are considered to be appropriate methods and tests. When the final plans have been drafted by the manufacturer with the support described by the FAA and with consideration of the inputs of the outside groups, this draft plan should be made available to airlines, federal research and evaluation authorities, and pilot groups for advance comment and coordination.

Checklist of Events

The following chart (figure A-1) summarizes the recommended sequential stages of implementation of FAR 25.1523. For each briefly described action, it is indicated when the procedures should be initiated and completed, who has the primary responsibility for planning and executing that step, and what rule applies.

<u>EVENT</u>	<u>TIME</u>	<u>RESPONSIBILITY</u>	<u>REFERENCE</u>
1. Aircraft design, including planning and conduct of design stage studies of crew workload	Program inception to specification of prototype	Manufacturer, (customer airlines and FAA)	
2. Decision as to general nature of FAR 25.1523 program New Model? Follow-on? Modification?	Prior to specification of prototype	FAA, (manufacturer and subsystem suppliers)	
3. Final flight deck design/prototype specification	Upon completion of necessary design studies/market analysis	Manufacturer*	
4. Preparation of overall plan for demonstration of compliance with FAR 25.1523	After prototype specification	Manufacturer, (FAA and interested outside parties)	8110.8, Paragraph 187
5. Constitution Type Certification Board	Upon receipt of application	FAA	8110.4, Paragraph 10
6. Review of applicant's data, plan, and issue identifications	Preliminary TCB meeting	FAA	8110.4, Paragraph 11
7. Preliminary decision on resolution of crew workload issues--identification of remaining issues	Upon completion of analysis of applicant's data	FAA	
8. Approval of plan (step 4) for overall compliance documentation	After study	FAA	

*Customer participation in all phases of flight deck evaluation, including minimum crew determination, is implicit in the manufacturer's responsibilities. Consultation between the manufacturer and customers is continuous from program inception through the phase of aircraft delivery, and until completion of aircraft service life.

9. Conduct of planned studies/simulations/analyses	From plan approval to initial aircraft production	Manufacturer, (FAA)	
10. MEL- Flight Manual		Manufacturer, (FAA)	
11. Decisions on requirements for flight test as part of functional and reliability (F&R) series	Upon completion of ground studies	FAA	8110.4, Paragraph 69
12. Conduct of flight tests (F&R)	From flight test plan to completion of tests	Manufacturer and FAA jointly	
13. Decision on requirements for further flight tests or mini-airlines series	After F&R series	FAA	
14. Conduct of final flight tests	After selecting routes, conditions; and pilot training	FAA (Manufacturer)	
15. Conduct of maintenance performance assurance tests	From step 9	FAA (Manufacturer)	
16. Conversion of records of all issues tests and conclusions in an active file	When all data is available	FAA	
17. Review total program — final certification decision	Final step	FAA	

**Development of the MEL parallels that of the flight manual. Based on design changes and proposed flight procedures, both are drafted in preparation for design stage engineering simulation. Both are further evolved as required by results of design and proving tests. The flight manual and MEL must be complete in proposed final before the start of functional and reliability flight testing.

APPENDIX

REFERENCES

1. FAA Order 8110.4, "Type Certification," Department of Transportation, Federal Aviation Administration, December 1967.
2. FAA Order 8110.8, "Engineering Flight Test Guide for Transport Category Airplanes," Department of Transportation, Federal Aviation Administration, September 1974.
3. Brown, E. L., G. Stone, and W. E. Pearce, "Improving Cockpits through Flight Crew Workload Measurements," Douglas Aircraft Company, Douglas Paper No. 6355, April 1975.
4. FAR Part 25, "Airworthiness Standards: Transport Category Airplanes," Department of Transportation, Federal Aviation Administration, June 1974.

SECTION SEVEN
ISSUES

TABLE OF CONTENTS

	Page
SECTION 7	7-1
7. ISSUES	7-1
7.1 Definition of Verification and Validation	7-1
7.2 Means of Showing Compliance With Air Systems and Worthiness Requirements	7-2
7.3 Function Criticality	7-2
7.3.1 Function Performance Requirements	7-2
7.4 Maintenance of Software	7-4
7.5 Regulatory Review Period	7-4
7.6 Documentation	7-5
7.6.1 Software Documentation Standards	7-5
7.7 Software Configuration Management Control	7-5
7.7.1 Support Software Configuration Control	7-5
7.7.2 Configuration Management/Control of the Operational Flight Program	7-5
7.8 Higher Order Language	7-8
7.9 Redundant Systems Software	7-8
7.10 Test Boundaires	7-8
7.11 Role of Models in Validation	7-8
7.11.1 Analytic Reliability Model Issues	7-8
7.11.2 Scope of Analytic Models	7-9
7.11.3 Verification and Validation of Models	7-9
7.11.4 Interpretation of Model Results	7-10
7.11.5 Numerical Accuracy	7-10
7.11.6 Data Validity	7-11
7-12 Role of Simulation and Testing in Validation	7-11
7-13 References	7-16

LIST OF ILLUSTRATIONS

Figure		Page
7-1	Example of Documentation Tree	7-6
7-2	Simulator-Based System Validation Process (Reference 44)	7-14

LIST OF TABLES

Table		Page
7-1	Quantitative Safety Requirements	7-3
7-2	Digital Flight Control-Basic Function Reliability (Reference 16)	7-4
7-3	System Documentation	7-7
7-4	Example of Iron Bird Simulation Use (Reference 43)	7-12
7-5	Characteristics to be Addressed in the Validation Process (Reference 44)	7-13
7-6	Areas of FAR 25 that are Prime Candidates for Increased Usage of Simulation for Certification (Reference 45)	7-15

SECTION 7

7. ISSUES

A variety of issues have arisen over the past decade with respect to validation and certification of digital systems. These issues have been discussed in references 1 through 11. The following paragraphs summarize some of the issues identified in these references.

7.1 DEFINITION OF VERIFICATION AND VALIDATION.

There has been no standard definition of verification and validation (references 8, 9, 10, and 12). Reference 2 defines validation, "loosely speaking as the demonstration that the systems meet rigorous reliability requirements." This reference further states that "the measurement of validation credibility, correctness, or assurance is illusive, much as is the measurement of a proof of Fermat's last theorem. In the final analysis, it is the judgment of an expert that matters. The validation must be measured from time to time in order to provide a credible basis for the acceptance of risk."

Reference 10 presents the definitions of verification and validation as:

"Verification - testing that an equipment manufacturer must perform to show that his product meets a specification.

"Validation - testing performed on an equipment, in conjunction with other equipment if necessary, to show that its specification was written correctly in the first place."

Reference 8 defines the terms as:

"Verification - The process of establishing that the developed software satisfies both the system requirements and the software requirements.

"Validation - The process of establishing that the delivered product, of which the software is a part, complies with aircraft level requirements; the higher-level follow-up to verification. Validation objectives are to demonstrate compliance with aircraft requirements under operational conditions and the absence of undesired effects. Validation procedures may include use of simulated aircraft characteristics and/or flight test."

Reference 12 defines the terms as:

"Verification is concerned with demonstrating the logical correctness and showing that the program performs according to its specifications.

"Validation is the process of demonstrating through testing in the real environment or an environment nearly as real as possible that the system not only is verifiable but also satisfies the user's requirements. In this sense, validation encompasses verification."

7.2 MEANS OF SHOWING COMPLIANCE WITH AIR SYSTEMS AND WORTHINESS REQUIREMENTS.

A major issue is the acceptable means of showing compliance with the system's airworthiness requirements. Reference 9 states "the regulatory authorities representatives were at pains to draw the distinction between systems' airworthiness requirement and means of showing compliance with these requirements." Software validation and verification was part of the latter.

7.3 FUNCTION CRITICALITY.

There has been and continues to be a great deal of discussion as to how to determine which functions are critical, essential, or nonessential. Reference 8 states that the System Requirements Document defines the criticality of the functions to be certificated. Reference 9 stated that an airline representative "urged software designers to make clear distinctions between critical and noncritical functions." It further states that there is "little justification for requiring software validation and verification for nonflight critical systems."

A serious question is the impact of function criticality on software certification needs (reference 10). The question was raised concerning the criticality categorization for an equipment of system containing both critical and noncritical functions. The proposal that the category for the most critical function should be used was not acceptable to most committee members. They believe the software modules associated with specific functions should be noninteractive and that each function should be categorized differently. Modifications could then be made to low-criticality function software modules without affecting the certification of the higher-criticality modules (reference 10). Reference 10 further identifies the need to "develop guidance on how to differentiate between critical and noncritical functions on the basis of the effects of their losses on the aircraft."

7.3.1 Function Performance Requirements.

7.3.1.1 Mission Accomplishment Reliability. The probability of mission failure due to failures in the flight control and avionics system is often stated as a requirement by the procuring company or agency. The USAF requires this value to be either determined by the use of a specified overall aircraft mission accomplishment reliability (R_M) and an allocation factor for flight control, or be $\leq 1 \times 10^{-3}$ where R_M is not specified (reference 13).

7.3.1.2 Quantitative Flight Safety. The probability of aircraft loss per flight due to failures in the flight control and avionics systems is often stated by the procuring company or agency. When this value is not specified, reference to either the value of 1×10^{-9} per hour (reference 14) or the values by aircraft type in MIL-F-9490D (Reference 13) are often used. Table 7-1 summarizes values from various sources. There is controversy as to what the value should be. Reference 5 states "In the domain of the flight critical, safety and reliability converge. In this region, the concepts of fail-passive and soft-disconnect do not apply. System failure cannot be tolerated, and system disconnect implies loss of control."

TABLE 7-1. QUANTITATIVE SAFETY REQUIREMENTS

Agency	Definition	Quantitative Value
USAF	Extremely Remote ¹	Class III Aircraft: $Q_s \leq 5 \times 10^{-7}$ per flight
	Probability of Aircraft Loss	Rotary Wing Aircraft: $Q_s \leq 25 \times 10^{-7}$ per flight
		Class I, II, & IV Aircraft: $Q_s \leq 100 \times 10^{-7}$ per flight
FAA	Extremely Improbable ²	1×10^{-9} or less per hour of flight time
	Probability of Aircraft Loss	
British Air Registration Board	Average Risk of a Fatal Accident ³	1×10^{-7} per landing
	Specific Risk of a Fatal Accident Take-Off	3×10^{-6} per landing
	Specific Risk of a Fatal Accident After Take-Off ³	3×10^{-6} per landing

¹Reference 13²Reference 14³Reference 17

"In order to achieve a satisfactory reliability and safety level, flight-critical DFCAS will utilize substantial hardware redundancy. The number of units of each type will depend on the reliability of individual units and the sophistication of the fault detection, isolation, and reconfiguration capability. For a DFCAS which is critical throughout the entire flight regime, an average hourly probability of failure not to exceed 1×10^{-9} is a foreseeable requirement. If the system architecture requires three operating computers to carry out signal selection and output voting, a minimum of five computers would be required to obtain an acceptable level of protection from random hardware failures, assuming a mean-time-between-random-hardware-failures of 10,000 hours, which is quite optimistic." (Reference 5)

Other studies (references 15 and 16) indicate that the system safety requirements of 10^{-7} to 10^{-9} per flight hour cannot be met by any of the digital flight control system architectures using realistic failure rates. Table 7-2 presents a synopsis of the analyses performed in reference 16.

Work performed under NASA contracts on a Fault-Tolerant Multiprocessor (FTMP) (reference 18) and a Software-Implemented Fault-Tolerance (SIFT) computer (reference 19) contained reliability calculations only for the computer units and not the sensors and actuators. Both references reported probabilities of failure of less than 10^{-9} after a 10-hour flight. Obviously, the results do not compare directly with those in table 7-2. The values used for the Mean Time Between Failure (MTBF) for similar components also do not agree in the three references (references 16, 18, and 19). This issue must be considered in evaluating any reliability/safety calculations.

TABLE 7-2. DIGITAL FLIGHT CONTROL-BASIC FUNCTION
RELIABILITY (REFERENCE 16)

System Architecture	Redundancy Management	Probability of Loss of Function at	
		2 Hours	4 Hours
Dual Digital (Inter-Unit Selection) (.99 coverage)	Dual Unit On-Line	1.12×10^{-4}	2.2×10^{-4}
	Single Unit On-Line	6.0×10^{-5}	1.2×10^{-4}
Triplex	Actuation Servo Force Voting	6.6×10^{-5}	2.7×10^{-4}
	Actuation Servo Force Voting & Servo Command Voting	3.4×10^{-5}	1.35×10^{-4}
	Actuation Servo Force Voting & Sensor Signal Voting	2.4×10^{-5}	9.8×10^{-5}
	Actuation Servo Force Voting, Servo Command Voting, & Sensor Signal Voting	1.95×10^{-5}	7.8×10^{-5}
	Actuation Servo Force Voting, Cross Strapped Sensor Signal Voting	1.65×10^{-5}	7.0×10^{-5}
	Actuation Servo Force Voting, Servo Command Voting, Cross Strapped Sensor Signal Voting	1.05×10^{-5}	4.4×10^{-5}

7.3.1.3 Automatic Flight Control System (AFCS) Performance Requirements. Accuracy requirements for output functions producing data and signals used by actuators and other systems have been prescribed by ARINC (references 20-24). The FAA has also issued specific performance requirements for Category II and III systems and subsystems (references 25-28). USAF requirements for flight control system performance are given in references 13, 29, and 30.

While the values given for desired accuracy of outputs may differ from reference to reference, the principal issue is the method or approach to be used for verifying that digital systems and subsystems can meet these requirements. Analog systems have a continuous signal, whereas the digital signal resolution (least significant bit) is a trade-off based on word length of the computer and the maximum value of the variable represented. Digital systems also require checks that the rate of change of a variable does not exceed that possible in the physical world and a variable must be tested against the limits for the maximum value.

7.4 MAINTENANCE OF SOFTWARE.

Another issue is the maintenance and modification of software after it is validated and certified. The airlines "intend to make their own software changes as and when necessary" (reference 10). One airline said they did not expect to modify flight critical functions software (reference 10). Reference 11 discusses the "requirements for both the initial software certification of an equipment or system and those for recertification following the modifications which would be made to software during its service life-time."

7.5 REGULATORY REVIEW PERIOD.

"Whether or not regulatory authorities involvement in software should begin at the development stage" (reference 10) is an issue. "The question of a certification authority's need to review software design" (reference 10) is also an issue.

7.6 DOCUMENTATION.

A variety of documents are required for use in the validation and certification of advanced digital integrated flight control and avionics systems. The primary document categories are specifications, plans, and reports. Additional documentation is often needed as shown in the documentation tree example in figure 7-1. Table 7-3 lists some of the major documents in each category. All documentation should be referenced by an index, the Configuration Item Index. The procedure for assigning configuration item index labels should be explained in the Configuration Item Index document.

It should be noted that the documentation listed in table 7-3 includes system level documentation, hardware subsystem documentation, and software documentation. This list is more extensive than the "Software Documentation for Regulatory Approval" list presented in reference 8. The regulatory agency must be concerned with the total system in its analyses of requests for approval.

Appendix A of this handbook contains a detailed description of the recommended format and contents of the documentation given in table 7-3.

7.6.1 Software Documentation Standards.

The need for software documentation standards has been identified in a number of references (1 through 3 and 9). The exact format of the software documentation standards and the amount of detail for each document is an issue. Reference 11 states that equipment manufacturers said that "there were some items and suggested documentation standards of reference 8 that they would not release even if their customers wanted it!"

7.7 SOFTWARE CONFIGURATION MANAGEMENT CONTROL.

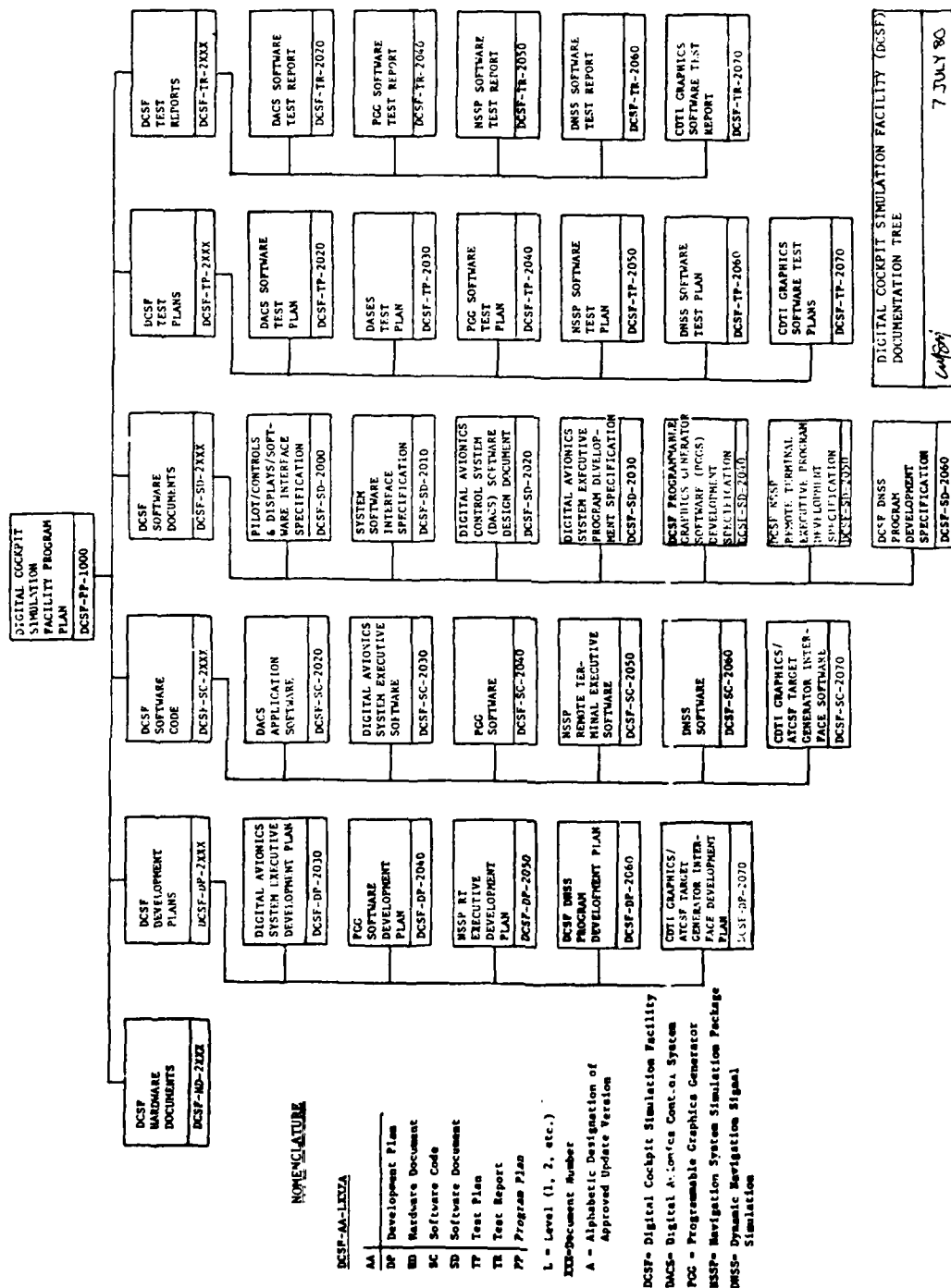
The need for software configuration management is clearly recognized (references 8, 31 through 35). The procedures for software management are an issue. These issues include support software management as well as the management of the software in the flight computers.

7.7.1 Support Software Configuration Control.

Reference 11 states that part number changes should occur "only when modifications (hardware or software) were made that affected equipment interchangeability." "The question next discussed was whether or not an equipment part number must change when the compiler used to generate the code with which that equipment was loaded was changed. It was suggested that a part number change would not be needed if the new compiler was shown to produce code that was bit-for-bit identical with the code with which the equipment was originally certified. If the new code was not bit-for-bit identical, then a review of the criticality of affected functions would be necessary before a decision was made" (reference 11).

7.7.2 Configuration Management/Control of the Operational Flight Program.

Reference 9 states "each memory component should have program identification in the first few memory locations." Reference 8 states "each preprogrammed memory device within a unit should be physically marked such that both hardware and software status may be identified." This implies that all code has an absolute address and



NOMENCLATURE

DCSF-AA-LEXXA

AA	Development Plan
HD	Hardware Document
SC	Software Code
SD	Software Document
TP	Test Plan
TR	Test Report
PP	Program Plan

L = Level (1, 2, etc.)

XX-Document Number
A = Alphabetic Designation of Approved Update Version

DCSF- Digital Cockpit Simulation Facility

DACS- Digital Avionics Control System

PGC - Programmable Graphics Generator

NSSP- Navigation System Simulation Package

DMS- Dynamic Navigation Signal Simulation

FIGURE 7-1. EXAMPLE OF DOCUMENTATION TREE

TABLE 7-3. SYSTEM DOCUMENTATION

<u>Document</u>	<u>Prepared By</u>	<u>Approved By</u>	<u>Used By</u>
System Requirements Description	Mfg*	C**	All
Configuration Item Index- Documentation Tree	Mfg	Mfg	All
Software Design Standards	Mfg	Mfg	All
Software Listings (Source Code)	Mfg	C, FAA	All
<u>Specifications</u>			
System Development Specification	Mfg	Mfg, C	All
System Product Specification	Mfg	All	All
System Software Development Specification	Mfg	Mfg, C	All
System Software Interface Specification	Mfg	All	All
Subsystem Computer Program Configuration Item (CPCI) Development Specification			
Subsystem Hardware	Mfg	C	C
Support Facility Development System Specification	Mfg	Mfg	Mfg
<u>Plans</u>			
Program Management Plan	Mfg	Mfg	Mfg
Configuration Management Plan	Mfg	Mfg	All
Software Development Plan	Mfg	Mfg	Mfg
Certification Plan	Mfg	FAA	Mfg, FAA
System Test Plan	Mfg	All	All
Subsystem CPCI Test Plan/Procedure	Mfg	All	All
<u>Reports</u>			
System Analysis Report	Mfg	Mfg	All
Software Stand-Alone Test Report	Mfg	Mfg	All
Software Integration Test Report	Mfg	Mfg	All
System Integration Report	Mfg	Mfg	All
Acceptance Test Report	Mfg	C	All
Operational Test & Evaluation Report	Mfg, C	C	All
<u>User Manuals</u>			
Programmer's Manual	Mfg	C	All
*Manufacturer **Customer			

that relocatable code will not be permitted. A question has been raised by fault tolerant system manufacturers as to the viability of this approach for fault tolerant system designs.

7.8 HIGHER ORDER LANGUAGE.

References 9 and 10 discuss the use of a standard higher order language for digital avionics programming and that "there was some support for this, given the known advantages of HUL" (reference 9). "A suggestion to specify one or more standard programming languages in order to ease the burden on certification authorities was rejected. The reason for the rejection was that to do so would interfere with designers' freedom of choice in an unprecedented manner" (reference 10).

7.9 REDUNDANT SYSTEMS SOFTWARE.

Reference 10 refers to "common mode problems, such as those that might result from the use of the same software in similar elements of dual or triple systems." Reference 11 states that "dissimilar software for each system of the redundant group was one technique" for protecting against common software errors in redundant systems.

7.10 TEST BOUNDARIES.

This issue deals with whether testing will be performed on equipment versus systems and whether software testing should go beyond testing of the function to include testing of the interface (reference 10).

7.11 ROLE OF MODELS IN VALIDATION.

Reference 11 states that the "report should define acceptable means for equipment manufacturers to show the regulatory authorities that safety is adequate." References 2 and 3 discuss in detail the role of analytical models in the validation process concerned with analyzing or predicting system performance or reliability.

7.11.1 Analytic Reliability Model Issues.

Reliability models are tools designed to evaluate the capability of a system to perform its assigned mission. They consider "reliability" — the rate at which failures occur — and "failure effects" — the results of specific failures. Missions for flight control systems are defined in terms of sensor, servo, and computational requirements. If the requirements are for minimum safe flight, then the reliability analysis becomes a safety analysis. The reliability models of interest in this section use some type of probabilistic approach (as opposed to Monte Carlo simulation).

Analytic reliability models offer several attractive features with regard to validation of digital flight control and avionics systems. They are much less expensive than bench tests, flight tests, and simulations. They can be applied early in the development of a system to document treatment of system reliability considerations. In addition, the models can provide a structure for numerical and expert assessment of system reliability. However, current analytic reliability models have some limitations and associated issues which must be considered. The following paragraphs discuss these factors.

7.11.2 Scope of Analytic Models.

The scope of useful analytic models is limited to a narrow range of system factors. Ideally, analytic models could handle the following classes of faults:

- Hardware operational fault.
- Hardware design faults.
- Software errors.
- Man-machine interface faults.

A number of methods address the first factor. Design faults, which may be important in very-large-scale-integrated circuits and in complex fault-tolerant approaches, apparently have not been included in any reliability models. Although a number of software reliability models exist, they are not well developed and none is known to be automated. In general, these models are based on the author's hypothesis of the relation among the original number of errors in the software, the number remaining, and the rate at which errors are detected and corrected. These models can be useful in large software development projects, but their credibility for predicting very low failure rates has not been established. In the man-machine interface area, some design principles have been developed through human factors research, but models for fault prediction and assessment appear to be nonexistent. As a result of these general limitations, current analytic models are restricted to hardware operational faults.

7.11.3 Verification and Validation of Models.

In order for a model to be useful in the assessment of a particular system, the model itself must produce correct answers (assuming meaningful input data are available). The model should be verified and validated.

Verification is the process of ensuring that a model satisfies its specifications. Validation, which is broader in scope, addresses the question of whether or not a model properly represents the real world. Complete validation requires the use of actual data from a real system which can be used to evaluate the model outputs. Until such data have been collected for implemented digital flight control systems, analytic models cannot be validated.

In spite of the difficulties associated with complete validation, it is possible to develop sufficient confidence in analytic models to justify their use in system certification. Models can be verified to ensure the computations are performed correctly and the logic is consistent with the specifications. A series of test cases can be used to evaluate particular capabilities of a model. The test cases should be solved by one or more alternate methods and the results compared with the model results. Automated models or analyses by hand can be used for the alternate methods.

A recent survey of automated tools for reliability and failure effects analysis (reference 36) found little evidence of verification and validation work. Of course, some verification tests may have been performed prior to the completion of model development but not recorded in any public reports on the model.

Several models have some accumulated experience which supports their correctness. ARIES has been used by over 50 graduate students at UCLA to investigate various

aspects of fault-tolerant systems. CARSRA was used by Boeing and Litton Industries to analyze flight control systems. In each case, CARSRA results were checked for reasonableness against hand computations (which used simplifying assumptions and were known to be slightly in error). In addition, Battelle evaluated CARSRA with a set of test cases for NASA-Ames Research Center. The CARE III model has been subjected to a peer review and is undergoing evaluation and testing by a contractor under the sponsorship of NASA-Langley Research Center. All of these activities contribute to the degree of confidence an analyst can have regarding the correctness of current reliability models.

7.11.4 Interpretation of Model Results.

Reliability models are tools to aid in the analysis of systems; they are not "analysis machines" which produce exact results when the button is pushed. Models can be used by an analyst to investigate the major causes of system reliability problems, the effects of particular faults, and the likelihood of critical events. However, the analyst must understand the inherent assumptions of the model to correctly interpret the results. This is particularly true when a model is applied to a system which it cannot accurately represent. For instance, the logic structure of communications among alternate computational paths may be too complex to be represented by a given model. By making judicious approximations, the analyst can capture the majority of the failures and their effects. An understanding of the model assumptions and theory can assist the analyst in developing approximations and interpreting results.

7.11.5 Numerical Accuracy.

Aircraft failure because of flight control system failure is required to be extremely improbable (FAR, Part 25) (reference 37). "Extremely improbable" is usually interpreted to mean the probability of catastrophic failure is on the order of 1×10^{-9} per hour (reference 14). Computations to this level of accuracy require special considerations.

Double-precision arithmetic may be required to avoid round-off errors which can occur in products and sums of probabilities close to zero and one and truncation errors in series approximations to transcendental functions. Equations should be written with accuracy in mind. For example, consider the equation for the probability of failure of N items in series:

$$Q = 1 - (1 - P)^N$$

where P is the failure probability of a single item. If P is very small, then $1 - P$ will be very close to one. To illustrate, if P is 0.00000025, then $1 - P$ is 0.99999975. Of the 8 digits used to express $1 - P$, only the rightmost two digits are actually being used to express the significant digits of P. The six 9's are, in effect, only being used to locate these significant digits (75) relative to the decimal point. The computer, however, treats the six 9's as the most significant digits. If N is large, raising $1 - P$ to the N power may not provide sufficient accuracy. In general, the loss of accuracy will be more severe the smaller the value of P. An alternate approach is to compute Q by the following algorithm, shown in FORTRAN:

$$Q = P$$

$$DO\ 5\ I = 2, N$$

$$5\ Q = Q + P - Q * P$$

This approach avoids the inaccuracy caused by subtracting the small number P from 1.

Reliability analysis of digital flight control systems requires rigorous numerical accuracy for validation of safe operation. Assuming sufficiently accurate data are available, the analysis technique should be capable of maintaining accuracy in the computations.

7.11.6 Data Validity.

Given the existence of valid reliability models, the usefulness of those models in the certification process is limited by the validity of the input data. The accuracy of such data as mean-time-between failures (MTBF) for components, mean duration of transient faults, and coverage probabilities limit the meaningfulness of the results. For instance, a ten-percent change in each of the preceding quantities for a particular system could cause a significant change in the system reliability prediction. Errors of 10 percent, 50 percent, or more can be expected using currently available data. It is therefore incumbent on the user of a reliability model to investigate potential variations in the input data. Results of sensitivity analysis should be incorporated in the reliability and safety assessments of a system.

7.12 ROLE OF SIMULATION AND TESTING IN VALIDATION.

Aircraft certification has progressed from demonstrations which were performed entirely during flight test to those in recent years which apply varying degrees of ground-based simulators. In fact, the FAA has reviewed research on the possibility of increased use of simulation as a means of certification of the aircraft and the systems providing the avionics, flight control, and safety of flight functions, and will evaluate the use of this type of data in the certification process.

FAA Advisory Circular 21-14 dated June 1975 states, "greater use of simulation would be cost effective and could improve the quality of certification . . . (simulation could be used) for aircraft structures and systems compliance for some or perhaps most of the flight demonstration." (Reference 38)

The following constitutes positive arguments for the support of increased usage of simulation in the certification of avionic systems. Simulation as a means of certification demonstration is vital to timely and economic deployment of future avionics. Validation methodologies for advanced digital avionics and flight control systems will depend heavily on simulation (references 39 and 40). Simulation also provides a means of effectively testing functions which cannot be effectively or safely accomplished during flight test, such as emergency or malfunction situations. Often, failure modes and effects analysis (FMEA) documentation is submitted as part of the certification documentation. Use of simulation can verify and lend more confidence, accuracy, and comprehensiveness to the FMEA (reference 41).

Simulation is used at various stages in the development and validation of new digital systems. In the early phases before a specific hardware subsystem may be available, the digital hardware may be simulated and stand-alone tests of the input/output interfaces performed. Various faults, such as a short circuit, or open circuit, can be simulated one-at-a-time. Multiple fault testing may then be conducted. In subsequent phases, stand-alone tests may be conducted using the actual hardware with its interface with other subsystems simulated. The actual hardware item may also be tested using simulated interfaces with other subsystems while being subjected to environmental tests prescribed in DO-160A (reference 47).

During the integration of subsystems, simulation facilities are used to provide the necessary test driver signals from subsystems which have not been integrated. As additional subsystems are integrated one-at-a-time, the simulation test driver software requires changes to reflect the stage of system integration.

Modern-day examples of the use of simulation in the validation process are numerous. Formal qualification testing of the operational flight program (OFP) of the F-16's Fire Control Computer was demonstrated to the Air Force using a real time closed-loop simulator (reference 42). Validation of the A-7E's OFP was done using the Avionics Simulation Facility at the Naval Weapons Center. Table 7-4 shows the percent of testing in a ground-based simulator or iron bird used on the NASA F-8/DFBW project. Flight tests in this case were used mainly to validate results of the iron bird simulator (reference 43).

TABLE 7-4. EXAMPLE OF IRON BIRD SIMULATION USE (Reference 43)

	<u>PERCENTAGE OF TESTING</u>	
	<u>IRON BIRD</u>	<u>AIRCRAFT</u>
o Software Verification	98	2 (Preflt Test)
o FMEA Demonstration	90	10
o System Integration	75	25
o Ground Resonance		100
o Pilot Familiarization	95	5
o EMI	90	10
o Lightning Susceptability		100

The issues concerning simulation are, (1) how much can be demonstrated with what amount of confidence? (2) Will simulation ever be used for 100 percent of the certification demonstration? (3) If not 100 percent, what percent would be considered relying too heavily upon simulation?

The major published conclusions are that simulation can never totally replace flight test of advanced digital avionics and flight control systems. However, increased usage of simulation will dominate future validation methodologies. Notice in table 7-5, that the test methods/approaches to all the specification provisions involve simulation (reference 44).

TABLE 7-5. CHARACTERISTICS TO BE ADDRESSED IN THE VALIDATION PROCESS (Reference 44)

SPECIFICATION PROVISIONS	RELEVANT CHARACTERISTICS	TEST METHODS/APPROACHES
Functional Design (System Architecture)	Interface Compatibility	Hot Bench Dynamic Simulation
	Redundancy Management	
System Performance (Functional Modes)	Flying Qualities	Pilot-in-the-loop Simulation/Operation
	Autopilot Modes	Coupled Time Histories
	Autoland Performance	Statistical Performance
	Active Control	Time Histories/Spectral Analyses
Failure Effects (Safety)	Automatic or Crew Response	Fault Insertion Simulation/Operation
	Degraded Operation	
External Influences (Worst Case Conditions)	Gusts & Windshears	System Simulation/Operation
	Low Visibility	
	Electromagnetic Interference	
	Traffic Congestion	
Human Factors (Crew Acceptance)	Pilot Workload	Pilot-in-the-loop Simulation
	Control/Displays	Pilot-in-the-loop Simulation/Operation
	Operational Procedures	

Reference 44 also presents a generic simulator-based system validation process, which is shown in figure 7-2. Since all the square boxes represent simulation functions, it is apparent that heavy dependence is placed on the use of simulators for validation.

Reference 45 investigated the possibility of increased use of simulation for certification and identified areas where certification is currently not done by simulation and where more simulation could be a benefit as shown in table 7-6.

A problem that is still present in the use of simulation is that, although simulation can reasonably validate a correctly operating machine, existing technologies are not considered adequate to cope with validation of highly flight-critical systems under faulted conditions (reference 46). More research and documented experience is necessary in this area.

As stated previously, simulation as a means of certifying avionic equipment is currently used with valid justification for increased usage. However, simulation will never totally replace flight test for all phases of compliance of certification requirements. Areas of concern such as validity and completeness of fault insertion simulation should direct improvements and future research in simulation.

"The Federal Aviation Administration's main concern in the certification process is that whatever simulation technique is used is a valid replica of what happens to an aircraft in actual operation. Although this may seem to some a simplistic statement, we are faced day-in and day-out with accepting simulation techniques, both fast-time and real-time, which have insufficient validation to back up the claims of their certification usefulness.

Amendment 25-23 of the Federal Aviation Regulations brought forth a rewrite of Section 25.1309. Essentially, this rule addresses development of more complex,

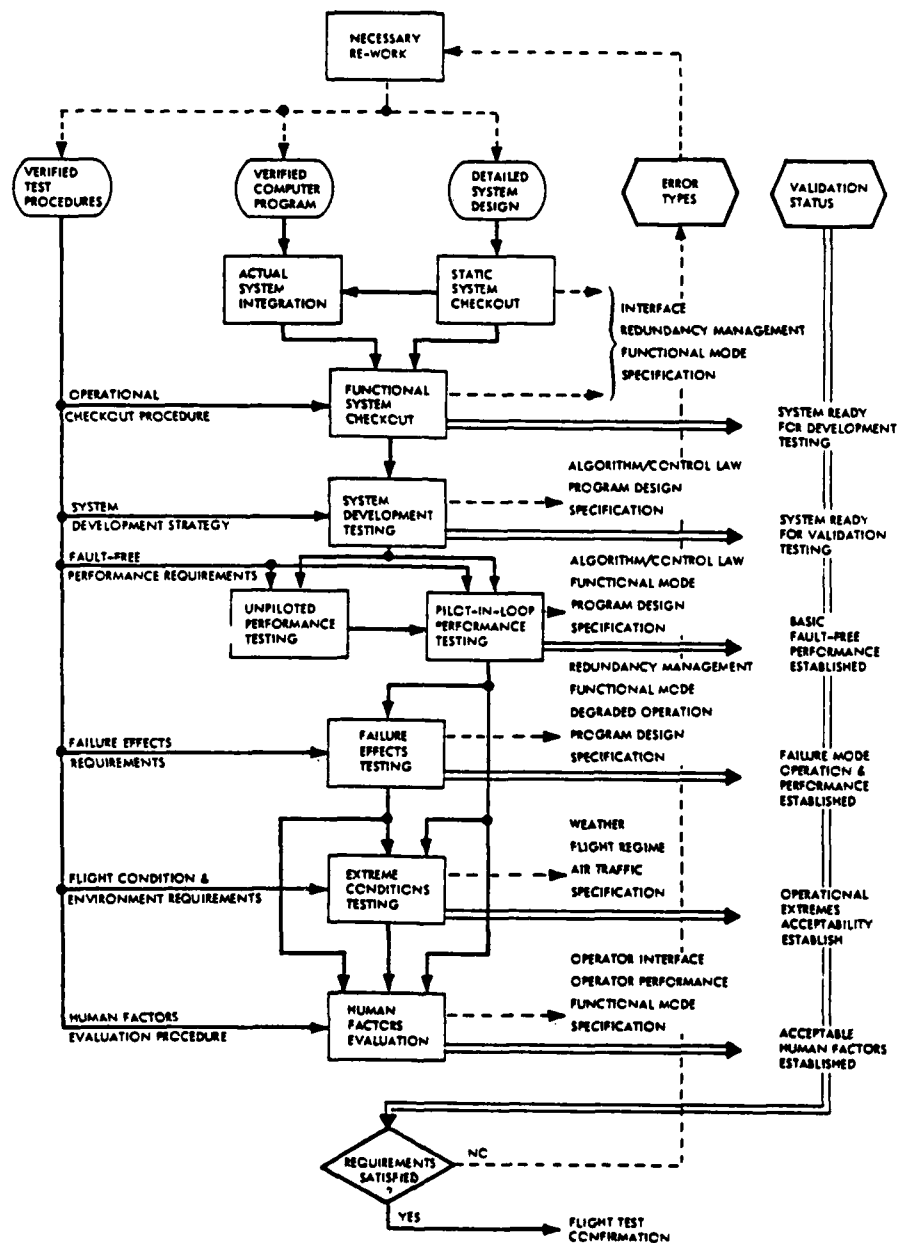


FIGURE 7-2. SIMULATOR-BASED SYSTEM VALIDATION PROCESS
(Reference 44)

TABLE 7-6. AREAS OF FAR 25 THAT ARE PRIME CANDIDATES FOR INCREASED
USAGE OF SIMULATION FOR CERTIFICATION (Reference 45)

FAR 25 Paragraph Number	Simulation Not Used; Use Justified	Function	Simulation is Used; Increase Justified	
			FAR 25 Paragraph Number	Function
25.105	Take-off		25.1301	Function and Installation
25.107	Take-off Speeds		25.1303	Flight and Navigation Instruments
25.109	Accelerate Stop Distance		25.1309	Equipment Systems and Installation
25.111	Take-off Path		25.1321	Arrangement and Visibility
25.113	Take-off Distance and Take-off Run		25.1323	Airspeed Indicating System
25.115	Take-off Flight Path		25.1325	Static Pressure System
25.117	Climb: General		25.1327	Magnetic Direction Indicator
25.119	Landing Climb: All Engines Operating		25.1329	Automatic Pilot System
25.121	Climb: One Engine Inoperative		25.1331	Instruments Using a Power Supply
25.123	Enroute Flight Paths		25.1419	Ice Protection
25.125	Landing		25.1431	Electronic Equipment
25.149	Minimum Control Speed		25.1457	Cockpit Voice Recorders
25.173	Static Longitudinal Stability		25.1459	Flight Recorders
25.175	Demonstration of Static Longitudinal Stability			
25.177	Static Directional and Lateral Stability			
25.181	Dynamic Longitudinal, Directional and Lateral Stability			
25.253	High Speed Characteristic			

interactive systems and the increasing criticality of these on the conduct of a "safe flight and landing." Here we must consider single and multiple faults in the system under study. Analytical tools frequently must be combined with technological judgments by FAA's pilots and engineers in order to find compliance. Use of simulation techniques, both real-time and fast-time, help greatly to build the analytical base to help make these decisions.

The use of the sophisticated flight simulator can be of great help in determining if continued safe flight and landing can be accomplished during failure conditions which are not extremely improbable. As required by Section 25.1309, we say, "for these failure conditions," a flight demonstration by the applicant in an airplane or satisfactory flight simulations of the worst-case failure conditions identified by the analysis may be necessary. After the demonstration of controlled flight under worst-case conditions for a period long enough to ensure that all of the consequences of the failure conditions have been experienced, the capability of accomplishing a safe landing on an airport must be demonstrated.

Here we have the "guts" of the FAA's requirement on the industry. Demonstrate to us that your system simulators, both fast-time and real-time, are valid, interface the man with the machine in these simulators, and then, after you build the machine, do sufficient validation flight testing to prove it out. Actual flight testing may become less and less a burdensome task as we gain more and more confidence in the various simulation techniques and environments. This is one area where we already exercise a form of our Regulation by Objectives concept, i.e., the FAA sets the safety objective, while the "how" to meet the objective is left with industry" (Reference 48).

7.13 REFERENCES.

1. Government/Industry Workshop on Methods for the Certification of Digital Flight Controls and Avionics, NASA TMX-73, 174, October 1976.
2. Validation Methods for Fault-Tolerant Avionics and Control Systems - Working Group Meeting I, NASA Conference Publication 2114, March 12-14.
3. Validation Methods Research for Fault-Tolerant Avionics and Control Systems-Working Group Meeting II, NASA Conference Publication 2130, October 3-4, 1979.
4. Waterman, Hugh E., FAA's Certification Position on Advanced Avionics, Astronautics and Aeronautics, May 1978, Pages 49-51.
5. Mulcare, D. B., Ness, W. G., McCarty, J. M., et al, Industry Perspective on Simulation Methods and Research for Validation and Failure Effects Analysis of Advanced Digital Flight Control/Avionics, Final Report NASA CR-152234, Lockheed-Georgia Company, January 22, 1978.
6. Bridgman, Michael S., Hitt, Ellis F., and Kenney, Suzanne M., Evaluation of Methods for Reliability and Failure Effects Analysis of Advanced Digital Flight Control and Avionics Systems, Battelle-Columbus Laboratories, April 21, 1980.

7. Hitt, Ellis F., Digital Avionics Validation, Final Report DOT-FA-03-81-P-00532, Battelle-Columbus Laboratories, February 27, 1981.
8. Software Considerations in Airborne Systems and Equipment Certification, Document No. RTCA DO-178, Radio Technical Commission for Aeronautics, November 1981.
9. Featherstone, David H., Minutes of the First Meeting, Special Committee 145, Digital Avionics Software, RTCA Paper No. 220-80/SC-145-4, Radio Technical Commission for Aeronautics, August 18, 1981.
10. Featherstone, David H., Minutes of the Second Meeting, Special Committee 145, Digital Avionics Software, RTCA Paper No. 268-80/SC-145-13, Radio Technical Commission for Aeronautics, October 21, 1980.
11. Featherstone, David H., Minutes of the Third Meeting, Special Committee 145, Digital Avionics Software, RTCA Paper No. 36-81/SC-145-20, Radio Technical Commission for Aeronautics, February 9, 1981.
12. Hitt, Ellis F., Definition of Verification and Validation, Letter to FAA, ACT-340, Battelle-Columbus Laboratories, March 19, 1981.
13. Flight Control Systems-Design, Installation, and Test of Piloted Aircraft, General Specification for, MIL-F9490D (USAF), June 6, 1965.
14. FAR Guidance Material, Airplane System Design Analyses, Advisory Circular 25. 1309-1, Federal Aviation Administration, September 7, 1982.
15. Hitt, E. F., Bridgman, M. S., and Robinson, A. C., Comparative Analysis of Techniques for Evaluating the Effectiveness of Aircraft Computing Systems, NASA CR-159358, Battelle-Columbus Laboratories, April 1981.
16. Erwood, R. G., McCorkle, R. D., Parente, J. J., et al, Digital Flight Control Redundancy Management System Development Program, AFFDL-TR-79-3050, Vol. I and II, Boeing Aerospace Company, May 1979.
17. Airworthiness Requirements for Automatic Landing Including Automatic Landing in Restricted Visibility Down to Category 3, British Civil Airworthiness Requirements Paper No. 367, Issue 3 - Working Draft, June 1970, Air Registration Board.
18. Smith, T. B., Hopkins, A. L., Taylor, W., Ausrotas, R. A., Lala, J. H., Hanley, L. D., and Martin, J. H., A Fault-Tolerant Multiprocessor Architecture for Aircraft, Volume I, NASA CR-3010, The Charles Stark Draper Laboratory, Incorporated, July 1976.
19. Wensley, J. H., Goldberg, J., Green, M. W., Kautz, W. H., Mills, M. E., Shostak, R. E., Whiting-O'Keefe, P. M., and Zeidler, H. M., Design Study of Software Implemented Fault-Tolerance (SIFT) Computer, Interim Technical Report 1, SRI International, June 1978.

20. Air Transport Inertial Navigation System-INS, ARINC Characteristic No. 561-11, Aeronautical Radio, Incorporated, January 17, 1975.
21. Flight Control Computer System, ARINC Characteristic 701, Aeronautical Radio, Incorporated, March 1, 1979.
22. Flight Management Computer System, ARINC Characteristic 702 (Draft 3), Aeronautical Radio, Incorporated, October 31, 1978.
23. Thrust Control Computer System, ARINC Characteristic 703, Aeronautical Radio, Incorporated, March 1, 1979.
24. Design Guidance, Air Transport Automatic Flight Control System, ARINC Report No. 417, Aeronautical Radio, Incorporated, April 9, 1971.
25. Criteria for Approval of Category III Landing Weather Minima, Advisory Circular 120-28C, Federal Aviation Administration, May 13, 1982.
26. Automatic Landing Systems, Advisory Circular 20-57A, Federal Aviation Administration, January 12, 1971.
27. Criteria for Approval of Category II Landing Weather Minima, Advisory Circular 120-20, Federal Aviation Administration, June 6, 1966.
28. Automatic Pilot Systems Approval, Advisory Circular 25.1329-1A, Federal Aviation Administration, July 8, 1968.
29. Townsend, James L., and Raymond Eugene T., Background Information and User Guide for MIL-F-9490D Flight Control Systems, Design, Installation and Test of Piloted Aircraft, General Specification for, AFFDL-TR-74-116, The Boeing Company, Wichita Division, January 1975.
30. Moynes, John F., Appendix to 'Background Information and User Guide for MIL-F-9490D' AFFDL-TR-74-116, AFFDL-TR-74-116, Supplement 1, Northrop Corporation, Aircraft Group, January 1980.
31. Management Guide to Avionics Software Acquisition, Vol. II - Software Acquisition Process, ASD-TR-76-11, Volume II, Logicon, Incorporated, June 1976.
32. Jensen, Randall W., and Tonies, Charles C., Software Engineering, Prentice-Hall, Incorporated, Englewood Cliffs, NJ, 1979.
33. Bersoff, Edward H., Henderson, Vilas D., and Siegel, Stanley G., Software Configuration Management, an Investment in Product Integrity. Prentice-Hall, Inc., Englewood Cliffs, NJ 1980.
34. Deutsch, Michael S., Software Project Verification and Validation, IEEE Computer, April 1981, Pages 54-70.
35. Myers, Glenford J., The Art of Software Testing, John Wiley & Sons, Incorporated, New York, 1979.

36. Ness, W. G., McCrary, W. C., Bridgman, M. S., Hitt, E. F., and Kenney, S. M., Automated Reliability and Failure Effects Methods for Digital Flight Control and Avionic Systems, NASA CR-166148, Lockheed-Georgia Company and Battelle-Columbus Laboratories, March 1981.
37. Federal Aviation Regulations, Part 25, Airworthiness Standards - Transport Category Aircraft, Federal Aviation Administration.
38. FAA Advisory Circular 21-14, June 1975.
39. Mulcare, D. B., and Ness, W. G., An Assessment of and Approach to the Validation of Digital Flight Control Systems.
40. Spradlin, Richard E., The 757/767 Flight Management System Laboratory Test Program, 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.
41. Urling, Herbert R., DC-9 Super 80 Digital Flight Guidance System Simulation Techniques for Certification, Proceedings of the 1981 RTCA Assembly, Radio Technical Commission for Aeronautics, November 1981, p.55.
42. Teichgraeber, Dr. Richard D., and DeMoss, Dr. Dean M., Real-Time Simulation as a Tool for F-16 Operational Flight Program Development and Testing, Second Digital Avionics System Conference, AIAA Paper No. 77-1527.
43. Jarvis, Calvin R., and Szalar, Kenneth J., Ground and Flight Test Experience with a Triple Redundant Digital Fly-by-Wire Control System, Advanced Aerodynamics and Active Controls, NASA Conference Publication 2172, October 1980.
44. Mulcare, D. B., et al, Industry Perspective on Simulation Methods and Research for Validation and Failure Effects Analysis of Advanced Digital Flight Control/Avionics, NASA Contract NAS 2-9784, January 22, 1978.
45. Archibald, Don M., The Role of Simulation in the Aircraft Certification Process, Federal Aviation Administration Report FAA-RD-77-17, January 1977.
46. Muclare, D. B., Ness, W. G., and McCord, J. E., Simulator Investigation Plan for Digital Flight Controls Validation Technology, Lockheed-Georgia Company, Engineering Report No. LG81ER0126, Revised April 10, 1981.
47. Environmental Conditions and Test Procedures for Airborne Equipment, Document No. RTCA DO-160A, Radio Technical Commission for Aeronautics, January 1980.
48. Luffsey, Walter S., Innovation in Regulation, Proceedings of the 1981 RTCA Assembly, Radio Technical Commission for Aeronautics, p.221, November 1981.

SECTION EIGHT
CONCEPTS/METHODOLOGIES

SECTION EIGHT

TABLE OF CONTENTS

	Page
SECTION 8	8-1
8. CONCEPTS/METHODOLOGIES	8-1
8.1 Overview	8-1
8.2 Analogy Methods	8-1
8.2.1 Good Engineering Judgment	8-1
8.3 Analytic Models and Methods	8-1
8.3.1 Terminology	8-4
8.3.2 Scope of Analytic Models and Methods	8-4
8.3.3 Criteria for Consideration of Analytic Models and Methods	8-4
8.3.4 Classes of Models and Methods	8-6
8.3.5 Role of Models and Methods in System Development	8-8
8.4 Software Verification/Testing	8-10
8.4.1 Life Cycle/Verification Activities	8-11
8.4.2 Manual Verification Methodologies	8-14
8.4.3 Computer-Aided Methodologies	8-17
8.4.4 Software Execution	8-24
8.4.5 Software Verification/Testing Conclusions	8-37
8.5 Hardware Verification/Testing	8-39
8.5.1 Bus Testing	8-39
8.5.2 ARINC 429-6	8-40
8.5.3 MIL-STD-1553/B	8-45
8.6 System Integration/Verification/Validation	8-53
8.6.1 Ground Simulation	8-53
8.6.2 Hot Bench	8-54
8.6.3 Iron Bird	8-55
8.6.4 Airborne Simulation	8-56
8.6.5 Flight Test	8-56
8.7 References	8-57

LIST OF ILLUSTRATIONS

Figure		Page
8-1	Methodologies Application in Validation of Digital Systems (2 Sheets)	8-2
8-2	Pitch Servo Fault Tree (Reference 3)	8-7
8-3	Typical Elements of an Automatic Test Analyzer (Reference 75)	8-23
8-4	429 Input/Output Circuit Standards	8-41
8-5	429 Output Signal Timing Tolerances	8-42
8-6	ARINC 429 Bus Voltages	8-43
8-7	Basic ARINC 429 Word	8-43
8-8	Stubs in a 429 Bus System	8-44
8-9	Data Encoding Manchester II	8-48
8-10	MIL-STD-1553B Word Format	8-49
8-11	Intermessage Gap and Response Time	8-50
8-12	Information Transfer Formats	8-50
8-13	Hot Bench Facility	8-54
8-14	Iron Bird Overview	8-55

LIST OF TABLES

Table		Page
8-1	Generation of Cut Sets (Reference 3)	8-9
8-2	Criteria for Evaluating Verification Methods	8-11
8-3	Life Cycle Verification Activities	8-12
8-4	Generic Heading for Static Tools and Techniques	8-19
8-5	Some Available Static and/or Dynamic Analyzers (2 Sheets)	8-20
8-6	Generic Headings for Dynamic Tools and Techniques	8-22

LIST OF TABLES (Continued)

Table		Page
8-7	Software Testing Philosophies	8-25
8-8	Module Integration Testing Strategies	8-30
8-9	Types of System Tests	8-36
8-10	Examples of Errors Detected by Each Verification Activity (Reference 86)	8-38
8-11	Parameter Variation Versus Impact on Word Error Rate Experienced on Space Shuttle Program	8-40
8-12	Characteristics of ARINC 429-6	8-42
8-13	Comparison of Data Bus Characteristics (2 Sheets)	8-46
8-14	Comparison of Terminal Characteristics (2 Sheets)	8-51

SECTION 8

8. KNOWN CONCEPTS/METHODOLOGIES

8.1 OVERVIEW.

This section discusses various concepts and methodologies proposed or used for validation of digital systems. Figure 8-1 illustrates the general sequence of application of these methodologies from the initial systems requirements formulation to the final validation.

8.2 ANALOGY METHODS.

The historical analogy method of verifying and validating a system involves having available a large data base systems of similar design or utilizing similar components. This approach is exemplified by the reliability calculations based upon MIL-HDBK-217C, in which the piece part failure rates for semiconductor devices are used to arrive at an overall estimate of the system reliability. Many reliability calculations are based upon the comparison with data from similar systems.

8.2.1 Good Engineering Judgment.

References 1 and 2 state, "the principle means of evaluating a proposed system has been, and will remain, good engineering judgment." This concept is based upon the free interchange of information between the manufacturer seeking certification and the regulatory agency personnel. Reference 1 suggests that the industry design engineers contact the appropriate FAA Aircraft Certification Office "as soon as the decision is made to proceed with certification, and preferably as soon as design concepts are formulated. At this stage, the design concepts can be explained to our engineers, who can advise you regarding the foreseeable problems in certification and also guide you regarding the FAA's requirements for data and testing." This reference further states, "But to support a contention that a specific failure condition is 'extremely improbable,' it is necessary to either include in the designs sufficient redundancy that known service experience allows a judicious decision in that regard, or to show by analysis, using acceptable basic reliability data and methods, that the expected system reliability is sufficient. A combination of service experience reliability and analysis will be used in most applications."

The use of "known service experience" indicates that good engineering judgment is partially based upon historical analogy and requires a data base. It also requires that the regulatory agency personnel have extensive experience with the types of systems to be certified.

8.3 ANALYTIC MODELS AND METHODS.

A number of analytic models and methods for reliability analysis currently exist. They exhibit a range of features and capabilities. No single technique has emerged as a generally accepted standard for reliability and safety analysis of digital flight control and avionics systems. The selection of a model or method for a given application depends on the characteristics of the system to be evaluated, the resources available to the analyst, and the point of the system in its development cycle. This section discusses criteria for selection of a model or method, general group models and methods, and their use in the development cycle. The discussion

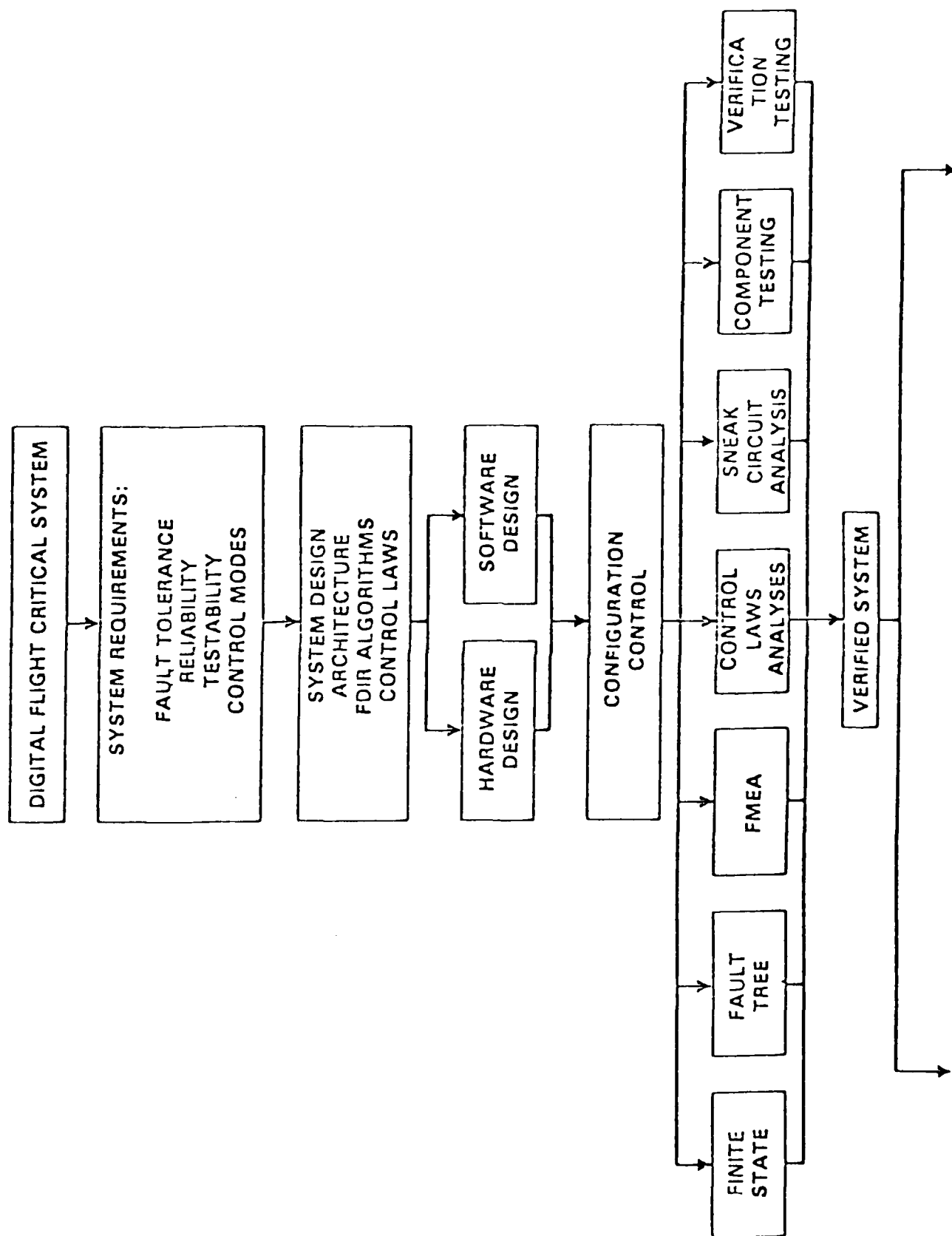


FIGURE 8-1. METHCDOLOGIES APPLICATION IN VALIDATION OF DIGITAL SYSTEMS

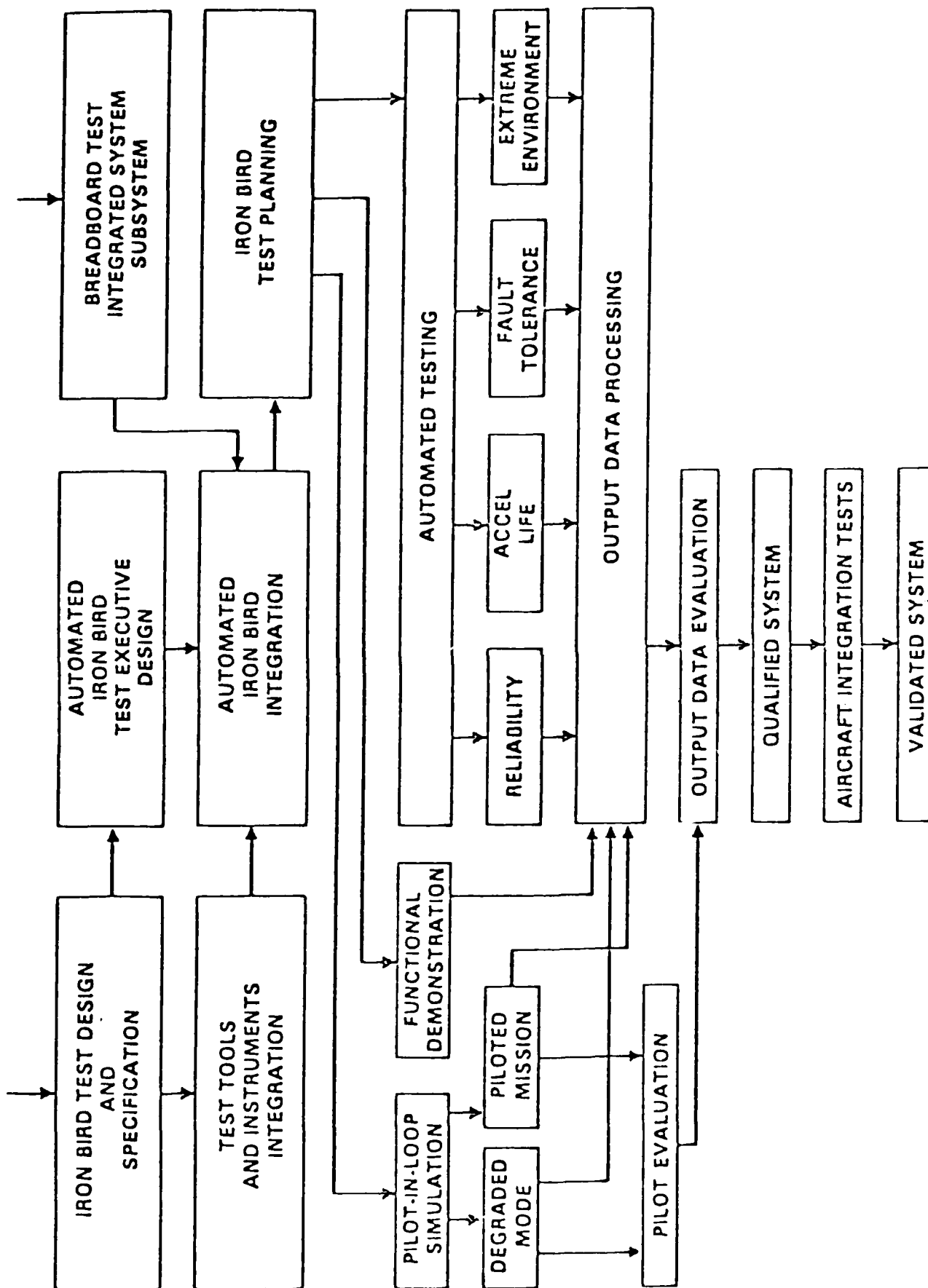


FIGURE 8-1 METHODOLOGIES APPLICATION IN VALIDATION OF DIGITAL SYSTEMS (Continued)

is limited to analytic models which address some aspect of reliability, failure effects, or safety of digital flight control and avionics systems.

8.3.1 Terminology.

"Reliability analysis" refers to evaluation of the capability of a system to perform its assigned mission. Missions of digital flight control and avionics systems are defined in terms of sensor, servo, display and computational requirements. The requirements may be the minimum for safe flight (in which case "reliability analysis" becomes "safety analysis") or they may be for some other level of performance. Reliability analysis is concerned with the rate at which failures occur and the impact of particular failures on system performance. "Reliability" refers to the probability of a failure occurring as a function of time, while "failure effects" are the results of specific failures.

8.3.2 Scope of Analytic Models and Methods.

As noted in section 7, current analytic models and methods are limited to consideration of hardware operational faults. The capabilities to model hardware design faults, software errors, and man-machine interface faults to the degree required for evaluation of fault tolerant systems have not been developed and validated.

8.3.3 Criteria for Consideration of Analytic Models and Methods.

The utility of an automated reliability model is a function of both its theoretical capabilities and its practical implementation. The model should be capable of producing valid and useful output for a variety of system designs. It should be capable of representing a range of system configuration characteristics and the potential types of faults. On the practical side, the model should be documented and available and should not place unreasonable demands on the user. The relative importance of the review criteria to a user is a function of the complexity of systems to be evaluated, the level of detail to be included, and the resources available to a user. Model preference is a selection of the relative weights placed on the criteria by the user. The following criteria were used in the evaluation reported in reference 3:

1. Required Output. The fundamental requirement of the model is to compute system reliability (i.e., the probability that the system operates successfully for time t). Implicit in this criterion is the capability to define system success in terms of system functions or structure so as to account for fault-tolerant capabilities and mission requirements. In addition, the model must maintain numerical accuracy consistent with outputs on the order of 10^{-3} to 10^{-11} (or the ranges specified in table 8-1, FAA AC 25:1309-1, or CAA Subsection D-1 (figure 1)).

2. Documentation. Understandable documentation for the model, such as a computer user's manual or programmer's manual, must be available if the model is to be applied correctly by the user.

3. Validation. The model should have been tested or accrued sufficient use in a variety of applications to create confidence in its ability to produce valid reliability estimates.

4. Computer Compatibility. Models are generally written in a single computer language and usually for a particular machine. These factors affect its transferability to other users.

5. Configuration Adaptability. Fault-tolerant system designs exhibit a wide range of architectures and mechanisms for fault detection, isolation, and recovery. To evaluate various configurations, models should be capable of handling the following classes of factors:

- . Computer design (e.g., failure modes, loss of subsets of functions)
- . Dependencies between subsystems
- . Number of redundant components

Redundancy management (e.g., active or standby spares, voting algorithms)

- . Permanent, latent, and transient faults
- . Fault recovery (i.e., coverage)
- . Data transmission schemes and bus architectures
- . Partitioning of processing functions

"Capable of handling" is to be understood to mean that the straightforward application of model capabilities can be used to express the impact of a factor on system reliability and failure effects.

Dependencies between subsystems can be particularly important in flight control systems. For example, failure of a sensor bus would cause the functional loss of all sensors which provide inputs only on that bus.

6. Output. In addition to system reliability, models should provide outputs which aid in understanding system performance. Examples of such outputs include the following:

The relative contribution of subsystems or modules of a system to system failure

Reliability improvement factor — the ratio of the unreliability of a baseline (i.e., without redundancy) system to the unreliability of the fault-tolerant version of the system

Numerical sensitivity

7. User Skills and Knowledge. All models require basic knowledge of reliability, engineering, and probability theory in addition to familiarity with the system to be analyzed. However, the skills and knowledge which must be possessed by a user to become familiar with a model should vary considerably from model to model. Models that are conceptually simple and easy to learn are obviously desirable.

8. Resource Requirements. Application of automated reliability models requires both staff time and computer resources. The resources required to prepare, execute, and analyze an individual computer run, as well as a variation of that run, are important factors in evaluating the relative utility of competing models.

These evaluation criteria may be different from the design criteria used for a model. Some models have been developed for particular types of systems or to study a particular aspect of fault tolerance. As a result, a model may be deficient for the current criteria even though it fully satisfied its own design goals.

8.3.4 Classes of Models and Methods.

The currently available analytic models and methods are grouped into three classes for discussion of their general capabilities and limitations. Specific tools which appear to have the most utility for evaluation of digital flight control and avionics systems are described in appendix B.

The three classes are:

- Reliability analysis models
- Fault trees
- Failure modes and effects analysis (FMEA).

8.3.4.1 Reliability Analysis Models. By the early 1970's, a number of reliability analysis models had been developed, including TASRA (reference 4), RBDCP (reference 5), REL 70 (reference 6), and REL COMP (reference 7). Each of these models is adequate for analysis of certain types of systems. However, the range of system types is so broad that none of these are adequate for all systems, especially those with more complex redundancy approaches and a large number of component types. Systems using redundant digital processors and memories are of this nature. Interest in such systems increased throughout the last decade since single processor and memory systems are unable to meet the reliability required for automatic landing and active flight control. More sophisticated models were developed to cope with predicting the reliability of more complex systems. By the late 1970's, ARIES (references 8 through 11), CARE II (references 12 through 13), CARSRA (references 14 through 18), CAST (references 19 through 20), and other models were available. As a group, these tools provided improved capability to represent more complex redundancy management schemes and transient faults. Continued evolution of models is evidenced by refinement and enhancement of ARIES and the development and testing of CARE III (reference 21). Reference 3 summarizes these models and presents a summary of the methods/criteria associated with each of the models.

8.3.4.2 Fault Trees. Fault trees are a generic method of evaluating the combinations of faults and external events or conditions which cause a specific top-level fault. The top-level event can be a system failure, in which case the lower-level faults are typically component faults. Alternatively, the top-level event could be at the component level, in which case the lower levels are usually piece-part faults. An inherent feature of fault trees is the graphic representation of the tree structure, a sample of which is shown in figure 8-2. The top event, Loss of Pitch Servo Function, will occur if there is a Loss of Control or Loss of Power. These two subevents are broken down into further detail by the events below them. Loss of Control occurs if either Mod Piston Jams or Loss of Both Coils occurs. The decomposition can continue until an appropriate level of detail is reached, and this level need not be the same for all branches.

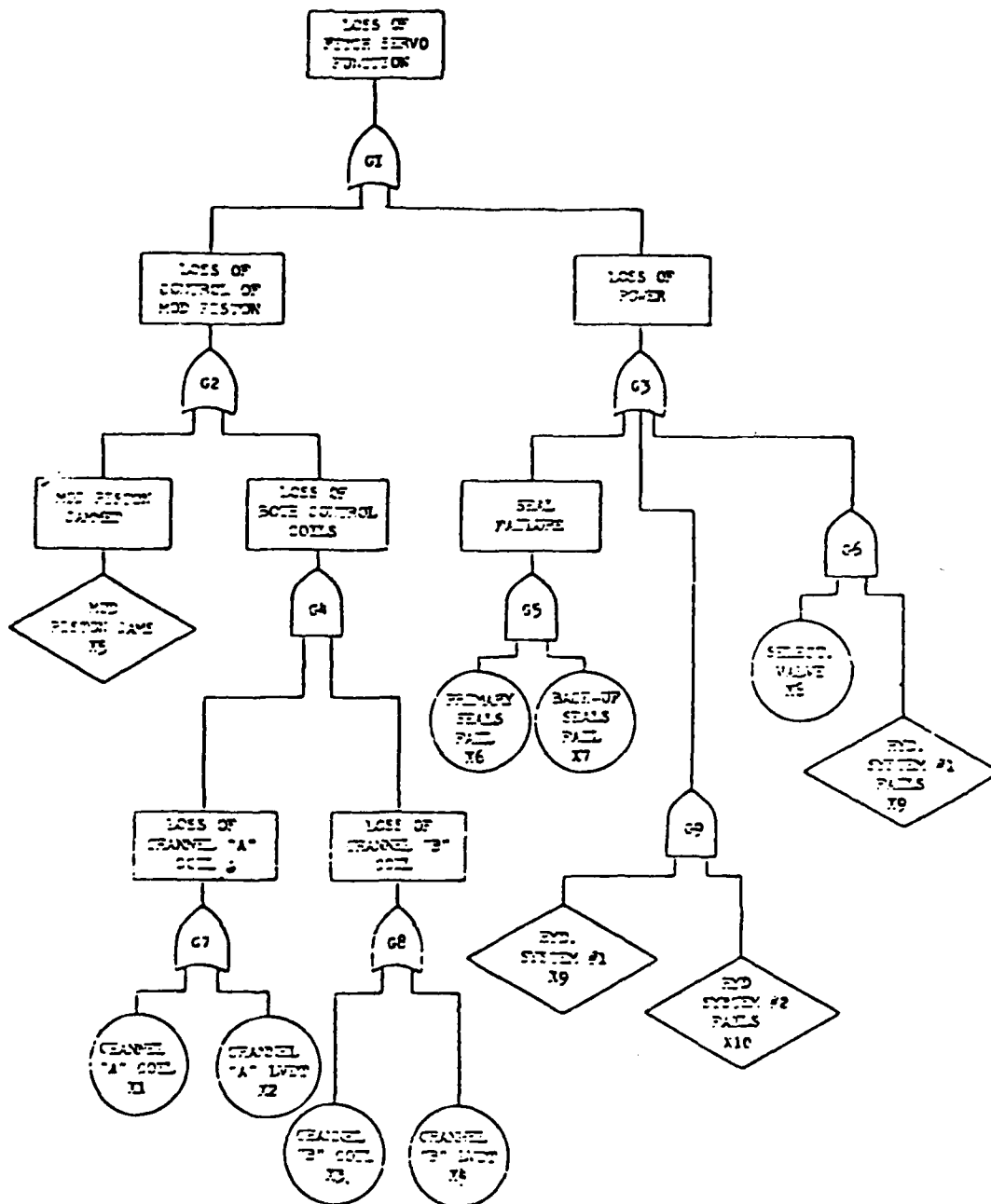


FIGURE 8-2. PITCH SERVO FAULT TREE (Reference 3)

Some symbology conventions have developed; G1, G2, G3, G7, and G8 are "OR" Gates, in that the occurrence of any input will result in the event immediately above the gate. G4, G5, G6, and G9 are "AND" Gates, so that all inputs are necessary for the event above the gate to occur.

For complex systems, the trees can get large, so that user aids to understanding the trees are essential. One such aid is the technique, either manual or automated, of obtaining the cut sets of the tree. These are the combinations of lowest-level events which will result in the top-level event. Table 8-1 illustrates one method of obtaining the cut sets of the simple tree under discussion. In this case, each cut set is minimal; that is, removal of any element results in the remaining elements no longer being a cut set.

The minimal cut sets are of interest from a qualitative point of view in that the lengths of the cut sets show the levels of redundancy of the design. The minimal cut sets are also of interest because they facilitate computation of the probability of the top-level event.

Probability computations begin with the input values for the probabilities of the lowest-level events. By combining these probabilities according to the logic of the fault tree, the probabilities of the next level of events are computed. The probability of the top-level event is found by continuing the process. The actual number of computations to reach the top level can be quite large. Minimal cut sets are used in some fault tree algorithms to reduce the computational effort.

Fault trees offer several important advantages. As a top-down documentation procedure, fault trees are useful guides for investigating the possible causes of system or subsystem failures. They are simple to learn. Failure rates are not limited to constant hazard rates since only a probability value is required for each of the lowest-level events. Various types of redundancy and dependencies among subsystems can be accommodated if the analyst is clever with conditional probabilities. Several automated fault trees are available to perform the probability computations.

Unfortunately, fault trees tend to grow in a nonlinear fashion as system size and complexity increase. They can quickly become too large to be practical when applied to flight control systems.

8.3.4.3 Failure Modes and Effects Analysis (FMEA). FMEA is a "bottom-up" approach for evaluating reliability aspects of the system design. The potential failure modes of each component or subsystem are identified. Each failure mode is then evaluated for its impacts on system performance. Probabilities can be assigned to failure modes to estimate the likelihood of failure effects. If criticalities or priorities are assigned to failure mode effects, then FMEA becomes failure modes, effects, and criticality analysis (FMECA).

FMEA provides a logical framework for reviewing an entire system. It can improve communication among design and evaluation personnel. Unfortunately, no automated tools for FMEA are known to exist.

8.3.5 Role of Models and Methods in System Development.

Currently available analytic models and methods can be used as tools to assist in the design, engineering development, and certification of digital flight control

TABLE 8-1. GENERATION OF CUT SETS (Reference 3)

STEP NO.	OLD CUT SET	NEW CUT SETS	COMMENTS
Initial	-	G1	Top Event Requires G1.
1	G1 }	G2 G3	Generate a new cut set for each input to OR Gate G1.
2	G2 { G3 }	X5 G4 G5 G6 G9	Generate a new cut set for each input to OR Gates G2 and G3.
3	X5 G4 G5 G6 G9	X5 G7 G8 X6 X7 X8 X9 X9 X10	Replace each AND Gate by all its inputs.
4	X5 G7 G8 { X6 X7 X8 X9 X9 X10	X5 X1 G8 X2 G8 X6 X7 X8 X9 X9 X10	Continue replacing gates by their inputs.
Final	X5 X1 G8 { X2 G8 { X6 X7 X8 X9 X9 X10	X5 X1 X3 X1 X2 X2 X3 X2 X4	Process terminates when all gates have been eliminated.

systems. All of these techniques are limited to operational faults of hardware. Assessment of system reliability requires assessment of hardware operational faults, design faults, software errors, and man-machine interface faults: Analytic models and methods can provide one aspect of the total equation.

8.3.5.1 Design Phase. In the design phase, models and methods can be used to evaluate the impacts of candidate system architectures and fault-tolerance techniques. Sensitivity analysis can be performed to assess the impacts of variations in levels of redundancy and in effectiveness of coverage. It will usually be sufficient to model coverage as a single parameter for each fault type of interest rather than model the components of coverage.

8.3.5.2 Engineering Development. As the system moves into engineering development, more model capabilities are required to better represent the operation of the hardware. More detailed investigation of recovery schemes and redundancy options may be used to assist engineers with design choices. If it is inadequate, analytic models and methods can help identify those subsets of the system which cause the most failures. Fault trees and FMEA can be used together to assist designers in identifying possible combinations of component failures which result in system failure and the effects of particular faults.

8.3.5.3 Certification. Analytic models and methods can support the certification process through more detailed analyses of the types noted above. Many input data values will be estimates. "Best case" and "worst case" analyses can be performed to determine bounds of system reliability for the ranges of the input values. The tools can be applied to individual subsystems of the flight control system. In addition, they can be applied to the entire system to evaluate interactions between subsystems and the use of functional redundancy.

It may be desirable to apply two or more models. No single analytic tool is capable of capturing all of the fault-tolerant characteristics of moderately complex systems. Application of different models and methods to the same system can result in slightly different estimates of system reliability. This effect can occur because of variations in the ways different tools handle small failure probabilities and their combinations. Furthermore, no current model has been thoroughly validated. Hence, the use of multiple techniques can provide increased confidence in the system reliability results.

8.4 SOFTWARE VERIFICATION/TESTING.

Reference 22 states, "A software verification methodology is an organized collection of work procedures, supported by appropriate tools and techniques, to establish the correctness of software implementation in a systematic and conclusive manner." Section 3 of the handbook begins the discussion of verification "work procedures" versus system life-cycle phase which will be continued along with other methodologies in the first category presented, Manual Verification Methodologies. "Appropriate tools and techniques" or the testing aspect of verification will be divided into two categories: Computer-Aided Methodologies and Software Execution. Note that simulation as a verification/testing tool is discussed below, while simulation as a validation method is discussed under System Integration/Verification/Validation in this section.

Testing is the basic, and by far the most widely used, verification mechanism, although it is not the only one. Testing is the process of executing a program with the intention of finding errors but it cannot demonstrate the absence of errors and is limited in its ability to demonstrate correctness (references 23 and 24). Therefore, testing can never truly verify a computer program but can only give the software developer increased confidence of the reliability of his product.

The user should familiarize himself with the verification methods criteria in table 8-2 in order to evaluate the methodologies presented (reference 22).

TABLE 8-2. CRITERIA FOR EVALUATING VERIFICATION METHODS

- . Identification and elimination of all errors at each stage, and ultimately, confirmation of their absence
- . Improved quality of test/simulation results, and consequent reduction in their duration
- . Self-documentation of verification results
- . Alleviation of tedious, routine, or error-prone tasks
- . Support of good software design/development practices
- . Generic value
- . Favorable cost/schedule impact

8.4.1 Life Cycle/Verification Activities

Reference 25 presents a concise, generally well supported description of verification as a parallel process throughout the life-cycle phases. This is a continuation of the material in section 3 on System Life Cycle. Below are excerpts from reference 25:

"Program testing, executing the software using representative data samples and comparing the actual results with the expected results, has been the fundamental technique used to determine errors as stated above. However, testing is difficult, time consuming, and inadequate. Consequently, increased emphasis has been placed upon insuring quality through the development process."

Table 8-3 shows each of the major life cycle phases and the corresponding verification activities. Each life cycle phase will be discussed in the table.

TABLE 8-3. LIFE CYCLE VERIFICATION ACTIVITIES

Life Cycle Phase	Verification Activities
Requirements	Determine Verification Approach Determine Adequacy of Requirements Generate Functional Test Data
Design	Determine Consistency of Design with Requirements Determine Adequacy of Design Generate Structural and Functional Test Data
Construction	Determine Consistency with Design Determine Adequacy of Implementation Generate Structural and Functional Test Data Apply Test Data
Operation and Maintenance	Retest

"Requirements. The verification activities that accompany the problem definition and requirements analysis stage of software development are extremely significant. The adequacy of the requirements must be thoroughly analyzed and initial test cases generated with the expected (correct) responses. Developing scenarios of expected system use may help to determine the test data and anticipated results. These tests will form the core of the final test set. Generating these tests and the expected behavior of the system clarifies the requirements and helps guarantee that they are testable. Requirements for which it is impossible to define test data or determine the expected value are ineffective and should be reformulated. Late discovery of requirements inadequacy can be very costly. A determination of the criticality of software quality attributes and the importance of validation should be made at this stage. Both product requirements and validation requirements should be established.

"Design. Organization of the verification effort and test management activities should be closely integrated with preliminary design. The general testing strategy, including test methods and test evaluation criteria, is formulated, and a test plan is produced. If the project size or criticality warrants, an independent test team is organized. In addition, a test schedule with observable milestones is constructed. At this same time, the framework for quality assurance and test documentation should be established (references 26-28).

"During detailed design, validation support tools should be acquired or developed and the test procedures themselves should be produced. Test data to exercise the functions introduced during the design process as well as test cases based upon the structure of the system should be generated. Thus, as the software development proceeds, a more effective set of test cases is built up.

"In addition to test organization and the generation of test cases to be used during construction, the design itself should be analyzed and examined for errors. Simulation can be used to verify properties of the system structures and subsystem interaction. Design walk-throughs should be used by the developers to verify the flow and logical structure of the system while design inspection should be performed by the test team. Missing cases, faulty logic, module interface mismatches, data structure inconsistencies, erroneous I/O assumptions, and user interface inadequacies are items of concern. The detailed design must be shown to be internally consistent, complete, and consistent with the preliminary design and requirements.

"Although much of the verification must be performed manually, the use of a formal design language can facilitate the analysis. Several different design methodologies are in current use. Top Down Design proposed by Harlan Mills of IBM (reference 29), Structured Design introduced by L. Constantine (reference 30), and the Jackson Method (reference 31) are examples. These techniques are manual and facilitate verification by providing a clear statement of the design (reference 32).

"Construction. Actual testing occurs during the construction stage of development. Many testing tools and techniques exist for this stage of system development. Code walk-through and code inspection are effective manual techniques. Static analysis techniques detect errors by analyzing program characteristics such as data flow and language construct usage. For programs of significant size, automated tools are required to perform this analysis. Dynamic analysis, performed as the code actually executes, is used to determine test coverage through various instrumentation techniques. Formal verification or proof techniques are used to provide further quality assurance.

"During the entire test process, careful control and management of test information is critical. Test sets, test results, and test reports should be catalogued and stored in a data base. For all but very small systems, automated tools are required to do an adequate job, for the bookkeeping chores alone become too large to be handled manually. A test driver, test data generation aids, test coverage tools, test results management aids, and report generators are usually required (reference 33).

"Maintenance. Over 50 percent of the life cycle costs of a software system are spent on maintenance. As the system is used, it is modified either to correct errors or to augment the original system. After each modification the system must be retested. Such retesting activity is termed regression testing. The goal of regression testing is to minimize the cost of system revalidation. Usually only those portions of the system impacted by the modifications are retested. However, changes at any level may necessitate retesting, reverifying and updating documentation at all levels below it. For example, a design change requires design reverification, unit retesting, and subsystem and system retesting. Test cases generated during system development are reused or used after appropriate modifications. The quality of the test documentation generated during system development and modified during maintenance will affect the cost of regression testing. If test data cases have been catalogued and preserved, duplication of effort will be minimized (reference 34)." (reference 25)

8.4.2 Manual Verification Methodologies.

The following manual verification techniques are applicable to all lifecycle phases although generally thought of mainly during the design and early construction phases. These methods include. Desk checking and review, structured walk-throughs and inspections, formal methods of proof of correctness, and symbolic execution. In the near future, more of these activities will have become automated (reference 25).

8.4.2.1 "Desk Checking and Peer Review. Desk checking is the most traditional means for analyzing a program. It is the foundation for the more disciplined techniques of walk-throughs, inspections, and reviews. In order to improve the effectiveness of desk checking, it is important that the programmer thoroughly review the problem definition and requirements, the design specification, the algorithms, and the code listings. In most instances, desk checking is used more as a debugging technique than a testing technique. Since seeing one's own errors is difficult; it is better if another person does the desk checking. For example, two programmers can trade listings and read each others code. This approach still lacks the group dynamics present in formal walk-throughs, inspections, and reviews.

"Another method, not directly involving testing, which tends to increase overall quality of software production, is peer review. There are a variety of implementations of peer review (reference 32), but all are based on a review of each programmer's code. A panel can be set up which reviews sample code on a regular basis for efficiency, style, adherence to standards, etc., and which provides feedback to the individual programmer. Another possibility is to maintain a notebook of required 'fixes' and revisions to the software and indicate the original programmer or designer. In a 'chief programmer team' (reference 33) environment, the librarian can collect data on programmer runs, error reports, etc., and act as a review board or pass the information on to a peer review panel.

8.4.2.2 "Walk-throughs, Inspections, and Reviews. Walk-throughs and inspections are formal manual techniques which are a natural evolution of desk checking. While both techniques share a common philosophy and similar organization, they are quite distinct in execution. Furthermore, while they both evolved from the simple desk check discipline of the single programmer, they use very disciplined procedures aimed at removing the major responsibility for verification from the developer.

"Both procedures require a team, usually directed by a moderator. The team includes the developer, but the remaining 3 to 6 members and the moderator should not be directly involved in the development effort. Both techniques are based on a reading of the product (e.g., requirements, specifications, or code) in a formal meeting environment with specific rules for evaluation. The difference between inspection and walk-through lies in the conduct of the meeting. Both methods require preparation and study by the team members, and scheduling and coordination by the team moderator.

"Inspection involves a step-by-step reading of the product, with each step checked against a predetermined list of criteria. These criteria include checks for historically common errors. Guidance for developing the test criteria can be found in references 32, 34, and 35. The developer is usually required to narrate the reading of the product. Many errors are found by the developer just by the simple act of reading aloud. Others, of course, are determined as a result of the discussion with team members and by applying the test criteria.

"Walk-throughs differ from inspections in that the programmer does not narrate a reading of the product by the team, but provides test data and leads the team through a manual simulation of the system. The test data are walked through the system, with intermediate results kept on a blackboard or paper. The test data should be kept simple given the constraints of human simulation. The purpose of the walk-through is to encourage discussion, not just to complete the system simulation on the test data. Most errors are discovered through questioning the developer's decisions at various stages, rather than through the application of the test data.

"At the problem definition stage, walk-through and inspection can be used to determine if the requirements satisfy the testability and adequacy measures as applicable to this stage in the development. If formal requirements are developed, formal methods, such as correctness techniques, may be applied to insure adherence with the quality factors.

"Walk-throughs and inspections should again be performed at the preliminary and detailed design stages. Design walk-throughs and inspections will be performed for each module and module interface. Adequacy and testability of the module interfaces are very important. Any changes which result from these analyses will cause at least a partial repetition of the verification at both stages and between the stages. A reexamination of the problem definition and requirements may also be required.

"Finally, the walk-through and inspection procedures should be performed on the code produced during the construction stage. Each module should be analyzed separately and as integrated parts of the finished software.

"Design reviews and audits are commonly performed as stages in software development. The Department of Defense has developed a standard audit and review procedure (reference 36) based on hardware procurement regulations. The process is representative of the use of formal reviews and includes:

- "1. System Requirements Review is an examination of the initial progress during the problem definition stage and of the convergence on a complete system configuration. Test planning and test documentation begin at this review.

2. System Design Review occurs when the system definition has reached a point where major system modules can be identified and completely specified along with the corresponding test requirements. The requirements for each major subsystem are examined along with the preliminary test plans. Tools required for verification support are identified and specified at this stage.

3. The Preliminary Design Review is a formal technical review of the basic design approach for each major subsystem or module. The revised requirements and preliminary design specifications for each major subsystem and all test plans, procedures, and documentation are reviewed at this stage. Development and verification tools are further identified at this stage. Changes in requirements will lead to an examination of the test requirements to maintain consistency.

4. The Critical Design Review occurs just prior to the beginning of the construction stage. The complete and detailed design specifications for each module and all draft test plans and documentation are examined. Again consistency with previous stages is reviewed, with particular attention given to determining if test plans and documentation reflect changes in the design specifications at all levels.

5. Two audits, the Functional Configuration Audit and the Physical Configuration Audit, are performed. The former determines if the subsystem performance meets the requirements. The latter audit is an examination of the actual code. In both audits, detailed attention is given to the documentation, manuals, and other supporting material.

6. A Formal Qualification Review is performed to determine, through testing, that the final coded subsystem conforms with the final system specifications and requirements. It is essentially the subsystem acceptance test.

8.4.2.3 "Proof of Correctness Techniques. Proof techniques as methods of validation have been used since von Neumann's time. These techniques usually consist of validating the consistency of an output assertion (specification) with respect to a program (or requirements or design specification) and an input assertion (specification). In the case of programs, the assertions are statements about the program's variables. The program is proved if whenever the input assertion is true for particular values of variables and the program executes, it can be shown that the output assertion is true for the possibly changed values of the program's variables. The issue of termination is normally treated separately.

"There are two approaches to proof of correctness: formal proof and informal proof. A formal proof consists of developing a mathematical logic consisting of axioms and inference rules and defining a proof to be either a proof tree in the natural deduction style (reference 37) or to be a finite sequence of axioms and inference rules in the Hilbert-Ackermann style (reference 38). The statement to be proved is at the root of the proof tree or is the last object in the proof sequence. Since the formal proof logic must also "talk about" the domain of the program and the operators that occur in the program, a second mathematical logic must be employed. This second mathematical logic is usually not decidable.

"Most recent research in applying proof techniques to verification has concentrated on programs. The techniques apply, however, equally well to any level of the development life cycle where a formal representation or description exists. The GYPSY (reference 39) and HDM (reference 40) methodologies use proof techniques throughout the development stages. HDM, for example, has as a goal the formal proof of each level of development. Good summaries of program proving and correctness research are in references 41 and 42.

"Heuristics for proving programs formally are essential but are not yet well enough developed to allow the formal verification of a large class of programs. In lieu of applying heuristics to the program, some approaches to verification require that the programmer provide information, interactively, to the verification system in order that the proof be completed. Examples include Gerhart's AFFIRM (reference 43) and Constable's PL/CV (reference 44). Such information may include facts about the program's domain and operators or facts about the program's intended function.

"Informal proof techniques follow the logical reasoning behind the formal proof techniques but without the formal logical system. Often the less formal techniques are more palatable to the programmers. The complexity of informal proof ranges from simple checks such as array bounds not being exceeded, to complex logic chains showing noninterference of processes accessing common data. Informal proof techniques are always used implicitly by programmers. To make them explicit is similar to imposing disciplines, such as structured walk-through, on the programmer" (reference 25).

8.4.2.4 "Symbolic Execution. Symbolic execution is a method of symbolically defining data that force program paths to be executed. Instead of executing the program with actual data values, the variable names that hold the input values are used. Thus, all variable manipulations and decisions are made symbolically. As a consequence, all variables become string variables, all assignments become string assignments, and all decision points are indeterminate.

"The result of a symbolic execution is a large, complex expression. The expression can be decomposed and viewed as a tree structure where each leaf represents a path through the program. The symbolic values of each variable are known at every point within the tree and the branch points of the tree represent the decision points of the program. Every program path is represented in the tree, and every branch path is effectively taken.

"There are two major difficulties with using symbolic execution as a test set construction mechanism. The first is the combinatorial explosion inherent in the tree structure construction. The number of paths in the symbolic execution tree structure may grow as an exponential in the length of the program leading to serious computational difficulties. If the program has loops, then the symbolic execution tree structure is necessarily infinite. Usually, only a finite number of loop executions is required enabling a finite loop unwinding to be performed. The second difficulty is that the problem of determining whether the conjunct has values which satisfy it is undecidable even with restricted programming languages." (reference 25)

8.4.3 Computer-Aided Methodologies.

In the following discussion, computer-aided methods will encompass any selfexisting program which takes, as input, a target source program which is to be verified and outputs various results depending upon the purpose of the computer-aided program and the user's input. Included in the above defined methods are: Compilers, assemblers, automatic testing analyzers, and other software tools and techniques.

A compiler/assembler verifies that its particular language's constraints are followed and that the source program is syntactically correct. In the 60's and early 70's, there was much concern over the verification of the compiler itself. There was little confidence in new languages and their associated compilers. The use of only "mature" or time tested compilers was recommended. The problems associated with errors caused by unverified and "immature" compilers has led the military to standardize on one language, ADA.

Automatic testing tools are divided into two main categories, static and dynamic. Static tools analyze the structure of the code, but the code is not executed. Dynamic tools involve deriving a test plan, executing test cases, and evaluating the results.

AD-A133 222

VALIDATION OF DIGITAL SYSTEMS IN AVIONICS AND FLIGHT
CONTROL APPLICATIONS. (U) BATTELLE COLUMBUS LABS OH
E F HITT ET AL. JUL 83 DOT/FAR/CT-82/115

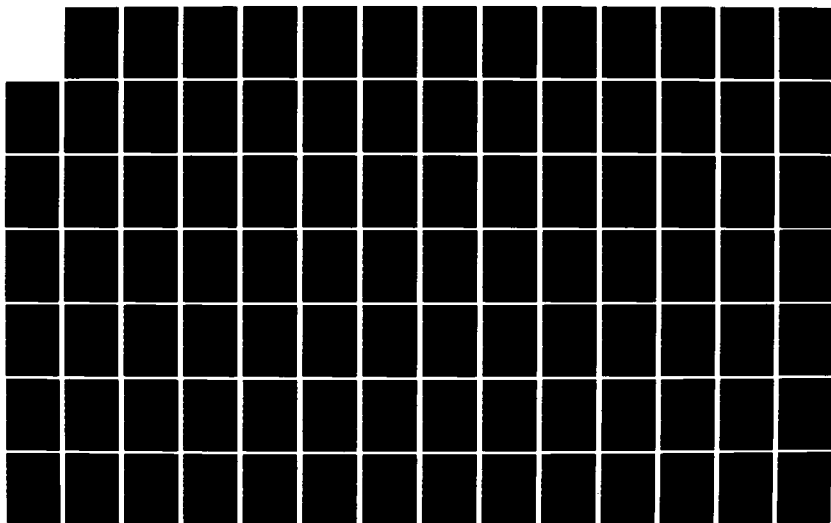
4/3

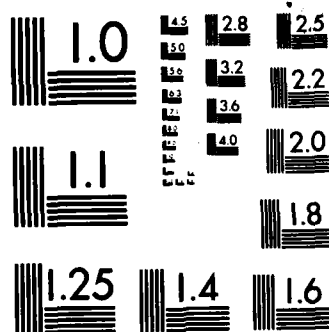
UNCLASSIFIED

DTFA03-81-C-00059

F/G 1/3

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

8.4.3.1 Static Analysis Techniques. In static analysis, information is obtained about the structure of the code which is particularly useful for discovering logical errors and questionable coding practices. Reference 45 details the types of information that can be obtained by static analysis (as stated in Section 3 but reproduced here for emphasis), which include (1) syntactic error messages; (2) number of occurrences of source statements by type; (3) cross-reference maps of identifier usage; (4) analysis of how the identifiers are used in each statement (data source, data sink, calling parameter, dummy parameter, subscript, etc.); (5) subroutines and functions called by each routine; (6) uninitialized variables; (7) variables set but not used; (8) isolated code segments that cannot be executed under any set of input data; (9) departures from coding standards (both language standards and local practice standards); and (10) misuse of global variables, common variables, and parameter lists (incorrect number of parameters, mismatched types, uninitialized input parameters, output parameters not assigned to, output parameters assigned but never used, parameters never used for either input or output, etc.).

Table 8-4 provides a list of the offered types of static tools and techniques along with their associated references - excerpted from reference 46.

Table 8-5 is a condensed list of existing static and/or dynamic analyzers derived from reference 67. A more comprehensive list can be obtained from the National Bureau of Standards' publication "NBS Software Tools Database" (reference 68).

Static analysis is very similar to certain phases of compilation, therefore computer science professionals feel that static analysis will be incorporated into future compilers as options (reference 5).

In dynamic testing and analysis by automatic test analyzers, the source code, in its original or modified form, is executed with test data. The only dynamic analysis technique that will guarantee the validity of a program is an exhaustive test, which is the test case of every possible element of the input domain. Unfortunately, the number of test cases would be, if not infinite, prohibitively large making this technique not practical (reference 46)

"The solution is to reduce this potentially infinite exhaustive testing process to a finite testing process. This is accomplished by finding criteria for choosing representative elements from the functional domain. These criteria may reflect either the functional description or the program structure.

8.4.3.2 Dynamic Analysis Techniques. Dynamic analysis is usually a three-step procedure involving static analysis and instrumentation of a program, execution of the instrumented program, and finally, analysis of the instrumentation data. Often this is accomplished interactively through automated tools.

"The simplest instrumentation technique for dynamic analysis is the insertion of a turnstyle or a counter. Branch or segment coverage and other such metrics are evaluated in this manner. A preprocessor analyzes the program (usually by generating a program graph) and inserts counters at the appropriate places.

TABLE 8-4. GENERIC HEADINGS FOR STATIC TOOLS AND TECHNIQUES

<u>TYPE</u>	<u>COMMENTS</u>	<u>REFERENCE(s)</u>
Accuracy Study Processor	determines calculated variable accuracy	47
Analyzer	provides indepth information on some feature of a program	48
Automated Test Generator	generates test data to transverse a particular logic path	49
Compiler	transforms higher order language to assembly of machine code while checking semantics	50-52
Consistency Checker	determines if requirements and/or designs are consistent	53
Correctness Proofs	derives axioms, theroms to prove program validity	54, 55
Cross-reference Program	used in testing and assessing impact of changes or other programs, parameter names, etc.	56
Design Language Processor	provides understandable representation of software design	57
Flowcharter	shows details of logical structure	58
Generator	produces test data or test cases	--
Language Processor	general category of compilers, assemblers	--
Map Program	provides location, size information	59
Modular Programming	method of producing small, functionally self-contained, interchangeable routines	--
Record Generator	generates test data relative to recorded formats	56
Software Monitor	provides detailed statistics about system performance	60,61,62
Standards Enforcer	checks procedures, rules and conventions of disciplined program design	63
Structure Analyzer	checks structure of control/data	64
Structured Programming	technique of using limited number of logic constructs	62,65,66
Test-Result Processor	provides output data reduction, formatting	--

TABLE 8-5. SOME AVAILABLE STATIC AND/OR DYNAMIC ANALYZERS

<u>Acronym or Abbreviation</u>	<u>Key Contact</u>	<u>Comments</u>
AFFIRM	USC Information Science Institute	Experimental methodology based on abstract data types, Pascal-like definition language.
AMPIC	Logicon	Structures, translates, executes programs, creates structured flowchart.
ARGUS MICRO	Boeing	Static & dynamic analysis, data flow diagram, viewfoils, utilities.
ATA	Science Applications	Dynamic verification via assertions. Output: Path usage, module tracing, statistics on correctness.
BEST/1	BGS Systems	Modeling package: Predicts response time, throughput. Planning analysis.
CA	TRW	Audits programs for adherence to 34 standards. Uses Process Construction Language (PCL).
COCOM	U.S. Air Force	Analysis, flowcharting, checks for conformity to coding standards.
C-PREP	Cogitronics	Preprocessor, update-management, (audit trail) text comparison, verification check.
DISSECT	U.C.S.D.	Experimental tool, evaluates execution runs when given execution paths & values.
DOCUMENTER-A	Softool	Language & compiler-independent, uses templates. Part of a tool system.
EAVS	General Research Corporation	Analyzes program, inserts probes, identifies untested paths.
FACES	COSMIC	Static analysis (structure, path, interface). Automatic interrogation routine.
FAME	H.O.S.	Interactive model analysis; consistency checks.
FFG	David Taylor Naval R&D Center	Experimental. Includes phases from conception to coding and automatic tests.
FORAN	U.S. Army	Static analysis on programs in any Fortran dialect. Flags syntax errors.
FTN. CODE ANAL.	U.S. Army	Audits programs for structured, optimized code; lists deviations.
GIRAFF	U.S. Air Force	Provides global name index, 24 reference categories.
INSTRUMENTER I,II	Softool	Generates executable profiles of programs; testing, optimization.
ISIS II TOOLBOX	Intel	Collection of utilities to improve productivity and reliability.
ITB	Software Research	"Interactive Test Bed" does QA, documentation, execution trace. Uses macros.
LOGIC-FLOW	Logicon	Uses Program Design Language (PDL). Analyzes syntax, logic, complexity. Flowcharter.
MSEF	Softech	Software definition & design for microcomputers under UNIX. Audit trail provided.
MTR	Caine Farber	Experimental; produces directed graphs, analyzes software systems.
N-SQUARED	U.S. Army	Produces traceability matrix from system description.
NODAL	TRW	Adds probes, analyzes execution, yield test effectiveness data.
PACE	TRW	Checks test coverage, static & dynamic analysis including probe generation.

TABLE 8-5. SOME AVAILABLE STATIC AND/OR DYNAMIC ANALYZERS
(Continued)

<u>Acronym or Abbreviation</u>	<u>Key Contact</u>	<u>Comments</u>
PDS	TRW	Tool set for structure, design; adds probes, yields statistics.
PET	McDonnell Douglas	Inserts probes, assertions. Analyses results, reports untested code.
PFORT	Bell Telephone	Checks compliance w. Fortran subset, gives crossreference on types, usage.
PSL/PSA	University of Michigan	Specifies & analyzes flow, structure, events check.
RADC FORT. ANAL.	Rome Air Development Center	Audits programs against Air Force Standards (design, structure etc).
RXVP-80	General Research Corporation	Automatic update, static structure analysis (module crossref., types, usage).
SAMM	Boeing	Based on activity cells. Model now automated. Uses integral hierarchy.
SAP/H	NASA/GSFC	Measures program complexity, SAP provides inputs.
SDS	U.S. Army	Effort to create total environment for real-time systems. Now in evaluation.
SDVS	Air Force Avionics Lab	Automatic simul. runs; configuration mgmt, allows software test w/o hardware.
SOFTOOL 80	Softtool	Struct'd. coding, prefab code library, configuration control, optimization, interface language.
SPEAR	General Dynamics	Automatic flowcharts, some checking of constructs.
STRUCT	U.S. Army	Audits programs for adherence to structured design. Summary output.
V-IFTRAN	General Research Corporation	Adds verification to IFTRAN's definition ability; uses executable assertions.
WELLMADE	Honeywell	Models abstract machines at different levels, from requirements to test phase.

"Instrumentation does not have to rely on direct code insertion. Often calls to run-time routines are inserted rather than actual counters. Some instrumented code is passed through a preprocessor/compiler which inserts the instrumentation only if certain commands are set to enable it.

"There are many other techniques for dynamic analysis. Most involve the dynamic (under execution) measurement of the behavior of a part of a program, where the features of interest have been isolated and instrumented based on a static analysis. Some typical techniques include expression analysis, flow analysis, and timing analysis" (reference 46).

Table 8-6 provides a list of the different types of dynamic tools and techniques along with their associated references (reference 46).

TABLE 8-6. GENERIC HEADINGS FOR DYNAMIC TOOLS AND TECHNIQUES

<u>TYPE</u>	<u>COMMENTS</u>	<u>REFERENCES</u>
Automated Verification Systems	instruments source code in order to determine coverage	64 69-71
Data Base Analyzer	provides information on data usage	72
Instruction Trace	provides a complete timed record of events during execution	--
Program Flow Analyzer	provides statistics on code usage and timing data	69, 70, 73
Program Sequencer	forces execution of all instructions and branches	74

(NOTE: Refer to table 8-5 for examples of existing static and /or dynamic analyzers.)

The user is once again referred to table 8-5 for examples of existing static and/or dynamic analyzers.

Figure 8-3 illustrates the typical elements of a full automatic test analyzer including both static and dynamic analysis (reference 73).

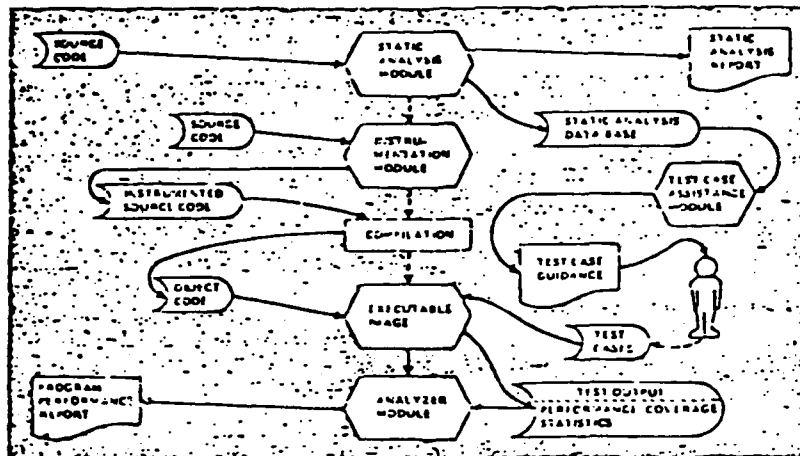


FIGURE 8-3. TYPICAL ELEMENTS OF AN AUTOMATIC TEST ANALYZER (Reference 75)

"The static analysis module analyzes the static structure of the code. Typically, this analysis would consist of partitioning each routine into path segments which support an evaluation of test thoroughness and analyzing the invocation structure of the program. This information is captured in a data-base file and formatted for output as a printed report.

"The instrumentation module acts as a preprocessor by inserting additional statements within the original source code. During execution, these statements or "probes" intercept the flow of execution at key points and record program performance statistics and signals in an intermediate file. The instrumented source code is compiled, an executable image is built, and the image is executed.

"The analyzer module functions as a postprocessor after the program execution. It formats and edits data recorded in an intermediate file during program execution and provides a printed report. This report furnishes information on test coverage, including:

- . Statements executed and frequency, percentage of statements executed
- . Statement sequences traversed, percentage of sequences traversed, listing of sequences not traversed
- . Data ranges of variables
- . Assertion violations.

The analyzer module may also compare anticipated test-case output augmented by any user-supplied evaluation criteria (prestored in a file) with actual output and list resulting discrepancies.

"The test case assistance module aids testing personnel in the selection of test inputs that will economically attain comprehensive testing goals. The module uses the static analysis data base and the coverage data recorded during execution to help testers in creating test cases of unexecuted code." (Reference 75)

8.4.4 Software Execution.

Of all the software testing and validation procedures available, one of the most often exercised methods is the testing process which essentially consists of executing the system software and accumulating performance statistics on its operation. In fact, according to reference 76, state-of-the-art verification and validation of programs are largely accomplished by exercising the software in a testing process. From the test results, proper functioning of the software and its reliability are inferred. However, it has been stated that this testing methodology does not result in a certified operational program. What is achieved is a validation that the programs have passed certain tests.

Reliability cannot be tested into a program; a program's reliability is established by the correctness of the design stages (reference 23). However, the probability of perfectly designing a large program is infinitesimal. Therefore, testing fills an important role in the overall system software development and certification.

This handbook section briefly describes some of the philosophies and strategies which are commonly employed during the testing and validation phase of software development when the software itself is being executed. Recommended validation and testing practices are presented in the Recommended Validation/Certification Practices Section of this text.

8.4.4.1 The Software Test Plan. Software testing encompasses a range of activities that are quite similar to the sequence of software development activities. These activities include establishing test objectives, designing test cases, writing test cases, testing test cases, executing the tests, and examining test case results. The successful engineering of a testing effort usually makes use of a great deal of accumulated software experience on the part of the participants. The prediction of expected program performance prior to testing has a scientific orientation, but the determination of what to test for, the design of specific tests, and the management of the test execution are artistic in nature. Although partly artistic in nature, testing is, however, not a "black magic" craft, but is a systematic and disciplined activity (reference 76). The purpose of the software test plan is to provide the control needed during the testing process.

The test plan defines the nature and scope of tests which are to be performed. Test procedures, keyed to the test plan, provide step-by-step instructions for the execution of the test and specify precisely what outputs are to be expected (reference 77). The test plan is initially produced early in the development cycle or during a definition phase prior to the manufacturing contract. It is then updated through progressive stages of the system (and software) development.

The test plan traces the testing sequence from unit level tests to final acceptance tests and identifies each individual test. This document is often accompanied by a "requirements-test matrix." This matrix identifies each individual requirement that is being tested and specifies the test or tests in which each requirement is to be verified (reference 76).

Each test indicated in the master test plan is given individual engineering attention; i.e., each test is "designed" and documented in test specifications. Observations of the test itself and evaluation of the test output data constitute the basis on which it is determined whether the test objectives have been met, the pertinent requirements verified, and the acceptance criteria satisfied. It should

be noted that the evaluation of the output data, when performed manually, is likely to be a tedious and time-consuming process for all but the most elementary of tests (reference 76). The manual task of error-checking is, in fact, in itself an error prone process (reference 30). This has been one of the major motivations for the development of automatic test tools whose utility and necessity are becoming increasingly paramount.

Disciplined control of the testing effort is maintained by emphasizing comprehensive and precise definitions of test plans, evaluation of test achievement against the test plan at periodic checkpoints, and quantitative measurement and expression of testing extent at checkpoints (reference 78). The importance of the test plan is independence of which type of software execution testing philosophy or strategy is being used. Some of these testing techniques are described in the following subsections.

8.4.4.2 Software Testing Philosophies. Of all the activities which make up software testing, test case design is by far the most crucial. In developing a philosophy for the design of test cases, there is a spectrum of opinions which range from testing to the specifications (without caring to look inside the code) to testing to the code (without caring to look inside the specifications) (reference 23). The type of testing philosophy chosen not only affects the test specifications described in the test plan document, but can also affect the type of software development used. The interaction between software testing and validation and software development is an ongoing process which has implications during each phase of the software development.

Table 8-7 shows a tabular representation of the extremes of some program software testing philosophies. Some key features of each philosophy are also shown. A more detailed explanation of the strategies can be found in the text which-follows.

TABLE 8-7. SOFTWARE TESTING PHILOSOPHIES

<u>Philosophy</u>	<u>Characteristics</u>	<u>Limitations</u>
Functional Testing	<ul style="list-style-type: none"> Design test cases by examining external specifications of the program Black-box approach Treat programs as functions Ultimate goal - test every possible combination and value of input 	<ul style="list-style-type: none"> Ignores important functional properties of the program not described in the requirements Difficult to determine test cases From practical viewpoint, impossible to test every possible input value
Structural Testing	<ul style="list-style-type: none"> Internal control structure of a program is used to guide the selection of test data Attempts to test every instruction in the code Stresses the implementation of the design requirements 	<ul style="list-style-type: none"> Difficult to test every path in large programs Although code may execute correctly, the wrong task may be performed Limited viewpoint

8.4.4.2.1 Program Size. It is intuitively reasonable to assert that a larger software package will require a more extensive testing effort than would a smaller package (assuming, hypothetically, that all other things are equal). Independent of other factors, the size of the eventual software end product is directly related to the complexity of the development both from a technical and managerial standpoint. This is illustrated by the consideration that a larger program almost invariably produces a larger number of critical logical paths that must be checked. The increased complexity associated with the growing number of contemporary large software projects has been another motivator for the development of automated software testing tools (reference 76).

The development of large quantities of software over short periods of time is often accomplished due to scheduling requirements. Such tight schedules compel the performing organization to apply a larger number of people to the problem. As more personnel become involved, the sensitivity to human fallibilities is magnified. There are more opportunities for mismatches of assumptions and logical reasoning to develop and assert a negative influence (reference 76). Thus the size of the finished software product must be considered when planning the testing effort.

8.4.4.2.2. Functional Testing. The end of the program software testing philosophy spectrum, which is concerned totally with designing test cases by examining the external specifications or interface specifications of the program (or routine) being tested, is referred to as functional testing (reference 23). With functional testing, the program is viewed as a black-box or a series of black-boxes. Tests are constructed from the functional properties of the program that are specified in the program's requirements. The ultimate goal of functional testing is to test every possible combination and value of input (reference 23).

In functional testing, a program (or routine) is considered to be a function (as in a mathematical function $a(x,y)$) and is thought of in terms of input values and corresponding output values. Programs usually have one or more input and one or more output values. Each variable is defined over a set of possible values called the domain of the variable. Domains can contain numbers, data structures, or even other programs. Functional testing requires that the selection of test data be made on the basis of the important properties of the elements in the domains of a program's input and output variables (reference 79).

The disadvantage of the black-box approach to software testing is that it ignores important functional properties of the programs which are part of its design or implementation and which are not described in the requirements (reference 79). The most obvious and generally intractable functional testing procedure is exhaustive testing. However, only a fraction of programs can be exhaustively tested since the domain of a program is usually infinite or infeasibly large and cannot be used as a test data set (reference 25). Thus, test data must be derived from an analysis of the functional requirements and include representative elements from all the variable domains. These data should include both valid and invalid inputs.

Generally, data in test data sets based on functional requirements analysis can be characterized as external, nonexternal, or special, depending on the source of their derivation (reference 25). The problem of deriving test data sets is to partition the program domain in some meaningful way so that input data sets which span the partition can be determined. Although there is no direct, easily stated procedure for forming this partition, it is agreed that the partitioning should be performed throughout the development life cycle (reference 25).

Reference 23 states that "testing is largely a problem in economics." Each test case should be designed to provide a maximum yield on the investment, where yield is measured by the probability that the test case will expose a previously undetected error and investment refers to the time and cost to produce, execute, and verify the test. Therefore, each test case should represent a class of inputs so that if the test case executes correctly, some confidence is gained that a certain class of inputs will execute correctly. However, doing this usually requires some knowledge of the logic and structure of the program, which tends to move away from functional testing towards the structural testing end of the testing philosophy spectrum (reference 23).

8.4.4.2.3 Structural Testing. Structural testing is an approach to testing in which the internal control structure of a program is used to guide the selection of test data (reference 79). One test case design criterion often used in structural testing is designing enough cases so that every software instruction is executed at least once. Another criterion is to design test cases so that every conditional branch instruction is exercised in every direction at least once. The ultimate goals of structural testing is to test every execution path through the program logic. The test cases are designed with little or no regard to the specifications (reference 23).

Structural testing is an attempt to take the internal functional properties of a program into account during test data generation and to avoid the limitations of black-box functional testing. Proposed methods of structural testing include branch testing and logical path testing. If the concept of a logical path is taken to mean any possible flow of control through a program, then the technique is impractical, as programs containing loops may have an infinite number of control paths (reference 79). One approach to the problem of too many paths is to group control paths into sets and to test one path from each set (reference 80). Another is to require that programs be constructed as a hierarchy of simple abstract procedures. The procedures should be small enough so that each control path through a procedure can be independently tested (reference 81).

Although used primarily during the coding phase, structural analysis should be used in all phases of the life cycle where the software is represented formally in some algorithmic, design, or requirements language. The intent of structural testing is to stress the implementation by finding test data that will force sufficient coverage of the structures present in the formal representation. In order to determine if the coverage is sufficient, it may be necessary to develop a structural coverage metric (reference 25). Thus the process of generating tests for structural testing is sometimes known as metric-based test data generation.

Metric-based test data generation can be divided into two categories based on the type of metric used: Complexity-based testing or coverage-based testing. In the latter, a criterion is used which provides a measure of the number of structural units of the software which are fully exercised by the test data sets. In the former category, tests are derived in proportion to the software complexity (reference 25).

It should be noted that even if all possible program paths could be checked, it is still quite possible that the test module does not perform its assigned task. It is possible that an incorrect function is implemented (such as a cube root instead of a square root) or that a required path is missing. A third subtle problem of structured testing is one of data sensitivities, where a path executes correctly for certain input data values but not for others (reference 23).

At this point, it should be noted that, in practice, neither extreme testing philosophy is actually applied without using some of the benefits of the other. A testing strategy is usually picked somewhere within the philosophy spectrum but somewhat toward the black-box functional testing end (reference 23). The strategy is then applied to the software throughout its development phases.

8.4.4.3 Software Testing Strategies. There is an almost infinite number of ways to implement and test any computer system. Although the design strategy currently used by most organizations tends to be a rather informal version of the top-down strategy — that is, the designer tries to design the major chunks of the system first, then breaks those chunks into smaller chunks, and so forth — the strategy used by many programmers to actually code the modules tends to be somewhat random (reference 30). It is sometimes argued that the testing strategy should be determined entirely by the coding strategy. However, although this correlation between coding strategy and testing strategy is common, it is not necessary (reference 30).

A software system is typically organized in a hierarchical structure and is composed of subsystems and lower-level components (references 23, 30, 76, and 77). The lowest-order component is denoted as a module or unit. The philosophy of testing at the unit level differs from that of integration testing where the units are interconnected to form higher-level components (reference 76). The following handbook subsections address module, integration, and overall system testing strategies involving actual execution of the software.

8.4.4.3.1 Module Level Testing. Module testing or unit testing is the verification of a single program module, usually in an isolated environment (reference 23). For all but the most simple software project, it is usual and prudent to approach testing in a progressive hierarchical manner, beginning at the software unit level. This is, in part, true because it is easier to test software exhaustively in small units (reference 76). The testing on this level emphasizes the verification of log', computations, data handling, timing, and sizing.

Reference 23 states that although test case design is a difficult process which involves both creative and artistic processes, there are a few simple rules that can be used to formulate a reasonable set of tests. These rules start by viewing the module as a black-box and using the module external specification to generate test cases and then progress to inspecting the insides of the module for supplemental test cases. The four steps are:

1. Using the module external specification, formulate a test case for each condition and option, the boundaries of all input domains and output ranges, and invalid conditions.
2. Inspect the code to ensure that all conditional branches will be executed in each direction.
3. Inspect the code to ensure that as many paths as feasible are covered with test cases.
4. Inspect the code for any sensitivities to particular input values and add test cases where necessary (reference 23).

One basic criterion for a set of test cases is ensuring that they cause every instruction in the module to be executed at least once. Reference 23 states that the minimum criteria for the unit test of any module is to execute all branches in all possible directions at least once. Several tools are available which help identify all module branches and paths. Flow charts, flow diagrams, and logic matrices are examples of such tools.

Once the test cases for the module have been designed, the next steps are the writing, testing, and execution of the test cases. When testing a single module in isolation, the test cases take the form of a module driver program. Each test case is represented by a call to the module, passing it unique sets of input data. An effort should be made to design the test cases so that they are self-checking, where the output checking is programmed into the test driver program (reference 23). This aids in reducing the number of module errors which slip through the testing process due to the driver outputs not being properly analyzed.

There are many types of errors which cannot be detected by just examining the module's outputs. These errors can be classified as erroneous side effects (reference 23). Module integration level testing is performed to expose this type of error condition.

8.4.4.3.2 Module Integration Level Testing. After successful unit-level testing, the units are connected to determine whether they function together in tandem. Integration testing is defined as the verification of the interfaces among system parts, such as modules, components, and subsystems (reference 23). Integration testing treats the software at the component level rather than at the detailed level of code that was the subject of unit testing. Thus, the main testing emphasis is on the interaction between software components and their interfaces (reference 92).

There are a large number of possible approaches that can be used to sequence the testing of modules and the merging of modules into larger entities. Most of the approaches can be described as variations of one of seven basic approaches. The seven approaches are: Bottom-up testing, top-down testing, modified top-down testing, big-bang testing, sandwich testing, modified sandwich testing (reference 23) and thread testing (reference 76). Table 8-8 gives some key points of each of the module integration testing strategies. A distinction is made in some texts between incremental and phased implementation methods also (references 76 and 30), but the seven techniques listed above encompass these methods. A description of each of the integration testing approaches follows.

8.4.4.3.2.1 Bottom-Up Testing. In the bottom-up approach, a program is merged and tested from the bottom to the top. The only modules that are module tested in isolation are the terminal modules (the ones that call no other ones). Once these modules are tested, using them can be assumed to be as reliable as using a built-up language function or an assignment statement. The next set of modules to be tested are the modules which directly call these tested routines. These higher-level modules are not tested in isolation, but are tested with the previously tested lower modules. This process is repeated until the top is reached, at which point the module test and integration test for the program would be complete (reference 23).

Bottom-up testing requires a module driver for each module, where the driver is a method of feeding test case input to the interface of the module being tested

TABLE 8-8. MODULE INTEGRATION TESTING STRATEGIES

<u>Method</u>	<u>Key Characteristics</u>
Bottom-Up Testing	<ul style="list-style-type: none"> • Program is merged and tested from the bottom to the top • Only terminal modules tested in isolation • Requires module drivers
Top-Down Testing	<ul style="list-style-type: none"> • Begins at the top of the program structure and proceeds to test components at progressively lower levels in the hierarchy • Requires dummy modules, or stubs • Combines module testing, integration testing, and a small amount of external function testing
Modified Top-Down Testing	<ul style="list-style-type: none"> • Same as top-down testing, but requires each module to be unit tested in isolation before it is integrated into the program
Big-Bang Testing	<ul style="list-style-type: none"> • Every module is unit tested in isolation • All modules integrated at once • Requires both stubs and module drivers • Debugging is difficult
Sandwich Testing	<ul style="list-style-type: none"> • Compromise between bottom-up and top-down testing • Testing begins from both top and bottom simultaneously, and proceeds until meeting in the middle • Difficult to apply to small programs • Requires both stubs and module drivers
Modified Sandwich Testing	<ul style="list-style-type: none"> • Modules in upper levels of the program are unit tested in isolation • Proceeds like Sandwich Testing
Thread Testing	<ul style="list-style-type: none"> • Useful in demonstrating the operation of certain important functions early in the testing process • Allows testing and analysis in digestible quantities • A thread is defined as a string of programs which accomplish a distinct processing function

(reference 23). A test driver can take one of two basic forms: A specialized driver or a "skeleton coding" of the superordinate. If skeleton coding is used to drive modules, the skeleton may be saved and used as the first cut on coding the actual superordinate when that stage is reached (reference 23), assuming that the code has not already been written. It should be noted that several automated module testers are available which help to make the testing procedure more controlled and reliable.

Perhaps the best justification for bottom-up development is the system whose low-level modules are critical in some sense. Another common justification for bottom-up development is based on scheduling of programmers, as bottom-up testing allows a program manager to assign large numbers of programmers to work, in parallel, on the bottom-level modules (reference 30). It should be noted that in real-life software development bottom-up testing is often used in combination with top-down testing, so that the advantages of each methodology can be applied to the system.

8.4.4.3.2.2 Top-Down Testing. The top-down approach to software testing begins at the top of the program structure and then proceeds to test components at progressively lower levels in the hierarchy (reference 76). It is important to see the interactions between levels in the hierarchy during top-down testing. At the time when any specific level is tested, the modules below the level must be specified, along with their interfaces to the level. However, these lower level modules have not yet been tested (and may not even have been written yet). Therefore, the lower level modules must be replaced with dummy modules, or stubs (reference 30).

The concept of a stub is an important aspect of top-down validation. In many cases, the dummy module can be totally absent. In other cases, the stub may just return a constant output. Often, the dummy modules are used to print messages so the software flow can be traced. In some cases, it may be appropriate to implement a primitive version of the actual module as the stub (reference 30).

The most significant advantage of top-down testing when compared to bottom-up testing is that it combines module testing, integration testing, and a small amount of external function testing. Related to this advantage is the advantage that test cases can be written as external inputs once the input/output modules are inserted. Top-down testing is also advantageous if the feasibility of the entire program is questionable or if there may be major flaws in the programmer's design (reference 23).

It is sometimes stated that the top-down approach tests the most important things first. However, it would be more accurate to say that, with top-down testing, different things are tested first. In some systems, the modules at the bottom of the hierarchy are critically important, and it could be advantageous to test them first (reference 30).

The ability to present users with an early version of the system is often claimed as another of the important benefits of top-down implementation. A skeleton version, containing stubs for many of the lower-level modules, can be demonstrated to the users to ensure that the programmers are implementing the system that was initially requested. In a sense then, top-down testing may compensate for inadequate problem specification or analysis, as the users are provided the opportunity to provide some feedback to the design process (they may have asked for certain features in the system without fully understanding the consequences) (reference 30).

A major disadvantage to top-down testing is that a module is rarely tested thoroughly right after it is integrated. A second subtle disadvantage of top-down testing is that it may lead to the belief that one can begin coding and testing the top of the program before the entire program is actually designed. Although this idea may initially sound economical, the iterative nature of program design usually causes just the opposite to be true (reference 23). Some of these weaknesses of top-down testing have led to the development of the modified top-down testing approach.

8.4.4.3.2.3 Modified Top-Down Testing. Modified top-down testing attempts to solve some of the problems associated with the top-down method by requiring that each module be unit tested in isolation before it is integrated into the program. The implementation of modified top-down testing would, therefore, require both module drivers and stubs for each module (reference 23).

The significant disadvantage of top-down testing, which is corrected by modified top-down testing, is its weakness in terms of thorough testing. Using top-down testing as previously described, it is often impossible to test certain logical conditions within the program such as error conditions or defensive programming checks. Top-down testing also makes checking of specific logical conditions within a module either difficult or impossible when the program is only using that module in a limited context. Even when testing of a condition is feasible, it is often difficult to determine what type of test case to write when the test case is entering the program at a point that is far removed from the condition (reference 23). Modified top-down testing avoids these problems of top-down testing.

8.4.4.3.2.4 Big-Bang Testing. In the big-bang method of testing, every module is first unit tested in isolation from every other module. After each module is tested, all of the modules are integrated together at once. The big-bang approach is probably the most common approach to the integration of modules (reference 23).

When compared to other testing approaches, big-bang testing has many disadvantages and few advantages. Both stubs and module drivers are required for every module. Modules are not integrated until late in the testing process, which could result in serious module interface errors remaining undetected for a long time. In both top-down and bottom-up integration testing, only a single module is being integrated at any point in time, so that when an error occurs, the most recently added module is the prime suspect. Debugging integration errors is more difficult in the big-bang approach (reference 23).

The big-bang testing approach does have some advantages. When the program being tested is small and well designed, this approach may be feasible. For large programs, however, the big-bang approach is usually disastrous (reference 23).

8.4.4.3.2.5 Sandwich Testing. Sandwich testing is a compromise between bottom-up and top-down testing which attempts to extract the advantages of each technique while eliminating some of the disadvantages.

In sandwich testing, the testing begins from the top-down and from the bottom-up simultaneously, and proceeds until eventually meeting somewhere in the middle. The point at which the convergence occurs depends on the particular program being tested. However, this point should be predetermined by examining the structure of the program (reference 23).

Sandwich testing is difficult to apply to small programs. It is a reasonable approach for the integration of a large program such as an operating system or an application system (reference 23).

Sandwich testing retains the bottom-up and top-down advantage of integrating modules together at an early point in the schedule. Because the top of the program is constructed early, one gains the top-down advantage of having an early skeletal working program. Because of the bottom-up construction of the lower levels of the program, the top-down testing problem of not being able to test specific conditions in the depth of the program is solved (reference 23). It should be noted that like the big-bang approach to module integration testing, the sandwich method requires the use of both stubs and module drivers. However, in sandwich testing, both are not required for every module.

8.4.4.3.2.6 Modified Sandwich Testing. Like top-down testing, sandwich testing has the problem, although in a smaller degree, of not being able to thoroughly test particular modules within the program. The bottom-up portion of sandwich testing solves this problem for the lower levels of modules within the program, but the problem may still exist for modules in the lower half of the top part of the program. In modified sandwich testing, the lower levels are still tested in a strictly bottom-up fashion. However, the modules in the upper levels of the program are first unit tested in isolation before they are integrated using top-down testing. Therefore, modified sandwich testing can be viewed as a compromise between bottom-up and modified top-down testing.

8.4.4.3.2.7 Thread Testing. Thread testing is a technique of functional testing that can demonstrate the operation of key functional capabilities fairly early in the testing activity. A thread is defined as a string of programs (or modules) which, when executed, accomplish a distinct processing function. When it is important to demonstrate the operation of certain important functions as early as possible, such as is often the case with real-time systems, thread testing can form the basis of the testing approach (reference 76).

Some of the benefits of the thread testing approach can be summarized as:

- . Allows testing and analysis in digestible quantities
- . Provides early demonstration of key functional capabilities
- . Forces the early availability of executable code
- . Requires early compliance with interface and configuration controls
- . Provides excellent visibility of status and quality of code
- . Produces early detailed design documentation

Utilization of thread testing requires that both top-down and bottom-up testing will be used, and implicitly determines the mix of the two approaches (reference 76).

Thread testing on a detailed level can form the basis of a very effective project planning and control strategy beginning at the point in the development cycle when

the requirements specification is delivered. The scheduling of thread demonstrations provides frequent identifiable milestones against which progress can be measured. This concept is attaining an increased popularity, particularly in connection with large software projects (reference 76).

8.4.4.3.3 Use of Simulation in Testing. Since it is usually not possible to test software under development in its actual operating environment using actual inputs, simulations are employed in the testing environment as a substitute for exercising the software in actual operations. The two classes of simulations used in the test environment are environmental simulation and interpretive computer simulation (reference 92). Brief descriptions of each class follows.

Environmental simulation, as the terminology suggests, feigns the environment in which the software will eventually perform. The environment includes other programs operating in parallel or in series, noncomputer hardware, and external inputs to the program (reference 82). Often, the simulator which is used in the testing can incorporate many useful debugging features that would probably not be available with the actual operational versions of the programs (reference 83).

With the advancing complexity and criticality of software functions, it is often necessary to control test inputs via simulation tools (reference 84). With particular reference to real-time systems, it has frequently become necessary to use simulation to generate a sufficient volume of test input data to stress the system. On interactive systems, simulators have been used to prepare scripts of input requests that normally would have been generated by operators at display consoles. Through the use of simulation, repeatability of inputs is attained that would not be possible by using real operators, consoles, and terminals (reference 76).

Interpretive computer simulation is employed to simulate the behavior of the operational computer when that machine is not available for testing the software. Recently, a tendency has been to simulate the instructions of the operational computer by using a microcoded program running in another computer (reference 82). Such simulations have been used to determine allocation of storage and to determine which programs should be kept in main memory and which should be kept in other storage devices (reference 76).

8.4.4.3.4 System Level Testing. The end objective of all testing and validation activities is to ensure that the delivered software product satisfies all specified functional and performance requirements and the identified design objectives (reference 85). If test cases are selected strictly with these high-level objectives in mind, then the set of test cases is not necessarily representative of the anticipated operational usage, in which case, reliability of the software is not necessarily demonstrated nor guaranteed. It is a frequent occurrence to find software errors during operational use that were not discovered during testing because no test case ever exercised certain sections of code (reference 76).

Reference 23 presents several "axioms" of testing, some of which are summarized below:

. A good test case is one that has a high probability of detecting an undiscovered error, not a test case that shows that the program works correctly.

- . One of the most difficult problems in testing is knowing when to stop.
- . It is impossible to test your own program.
- . A necessary part of every test case is a description of the expected output or results.
- . Avoid nonreproducible or on-the-fly testing.
- . Write test cases for invalid as well as valid input conditions.
- . Thoroughly inspect the results of each test.
- . As the number of detected errors in a piece of software increases, the probability of the existence of more undetected errors also increases.
- . Ensure that testability is a key objective to the software design.
- . The design of the system should be such that each module is integrated into the system only once.
- . The program should never be altered to make testing easier.
- . Testing must start with objectives (reference 23).

System testing is probably the most misunderstood form of testing. Function testing is the testing of all functions of the completely integrated system. System testing is the process of trying to find discrepancies between the system and its original objectives. System testing is a validation process when it is done in the end-user's actual environment. However, when the actual environment is not available, system testing is a verification process which is performed in a simulated or test environment (reference 23).

System testing often requires more creativity than the testing methods described previously. Designing good system test cases may require even more ingenuity than was required to design the system. Many of the types of tests which may be necessary during system testing are summarized in table 8-9. More detailed descriptions of the techniques are presented in the following paragraphs.

8.4.4.3.5 Load/Stress Testing. Load and stress testing attempts to subject the system to extreme pressures to expose errors that went undetected during light or moderate loads. Load/stress testing is often omitted because of the opinion that the system may never see the heavy loads in the real environment. However, this feeling is rarely correct (reference 23).

8.4.4.3.6 Volume Testing. While load/stress testing attempts to subject the system to heavy loads in a short period of time, volume testing attempts to expose the system to massive amounts of data over a longer period of time. The intent of volume testing is to show that the system or program can or cannot handle the amounts of data specified in the system objectives (reference 23).

TABLE 8-9. TYPES OF SYSTEM TESTS

<u>NAME</u>	<u>DESCRIPTION</u>
Load/Stress Testing	Subject the system to extreme pressures.
Volume Testing	Expose the system to massive amounts of data over a long period of time.
Configuration Testing	Test the software configuration.
Compatability Testing	If the system is used with an existing system, compatability testing is required.
Storage Testing	Tests main and secondary storage objectives.
Performance Testing	Tests performance of efficiency objectives.
Instability Testing	Tests the installation process of a system.
Reliability/Availability Testing	Shows that the system does or does not meet its original reliability.
Recovery Testing	Tests the recoverability of a system.
Serviceability Testing	Tests system serviceability and maintainability.
Human Factors Testing	Brings man into the testing loop.

8.4.4.3.7 Configuration Testing. If the software system can be configured, each software configuration must be tested. When the number of configurations is too large to allow for exhaustive testing, at the very least, the maximum and minimum configurations should be tested (reference 23).

8.4.4.3.8 Compatibility Testing. Many systems that are developed are improvements or additions to existing systems. In these cases, the system will have compatibility objectives which must be thoroughly tested (reference 23).

8.4.4.3.9 Storage Testing. Many systems have storage objectives stating the amounts of main and secondary storage used by the system under different conditions. Test cases should be written to attempt to check these conditions (reference 23).

8.4.4.3.10 Performance Testing. Performance or efficiency objectives such as response times and throughput rates under a variety of work loads and configurations are an important part of most system designs (reference 23). Test cases should be designed which check the various system performance criteria.

8.4.4.3.11 Installability Testing. Since many systems have complicated procedures for installation, testing the installation process of a system is vital (reference 23).

8.4.4.3.12 Reliability/Availability Testing. A key part of the system test is the attempt to show that the system does or does not meet its original reliability objects. Reliability testing is extremely difficult, yet efforts should be made to test as many of these objectives as possible (reference 23).

8.4.4.3.13 Recovery Testing. An important part of the objectives for some systems is a set of goals for system recovery functions. System testing is the proper time to attempt to show that these functions do work correctly (reference 23).

8.4.4.3.14 Serviceability Testing. When goals have been stated for the serviceability or maintainability characteristics of the system, all of the service aids, such as dump programs, trace programs, and diagnostic messages, must be tested during the system test. All maintenance procedures supplied with the system should also be tested (reference 23).

8.4.4.3.15 Human Factors Testing. Although some checking of the usability or human factors of the system should be done during the system test, this category is not as important as the other categories because it is normally too late to correct any major human factor problems. However, minor human factor problems, such as poorly worded system responses or messages, can often be detected and corrected during the system test (reference 23).

The basic rule in system testing is that "anything goes"; write highly destructive test cases, check the system at all of its operational boundaries, and write test cases representing possible user mistakes. Because of their nature, system test cases rarely involve the invocation of a single system function. System test cases are often written as scenarios representing a large set of sequential operations by a user. Because of their complexity, system test cases are composed of several parts: A script, the input data, and the expected outputs (reference 23).

It should be noted that as in module level and module integration testing, several automated system level software testing tools are available (references 23, 30, and 76). System testing is such a special and critical operation that in the future more automated tools will be developed and implemented to increase the effectiveness of the system test process.

8.4.5 Software Verification/Testing Conclusions.

In concluding the Software Verification/Testing section, the following items need to be stressed (very strongly), as they apply to digital avionics:

1. There exists no single methodology that covers all types of errors. Therefore, it is necessary to use a core set of software verification/testing tools and techniques in order to gain reasonable confidence in the software (reference 22).

2. Verification testing is not a function reserved for after coding is completed. Section 3, and the discussion under "manual methods" emphasized the requirement that verification/testing be throughout all life cycle phases.

3. The Department of Defense (DOD) has widely accepted the concept of Independent Verification and Validation (IV&V). The test objectives are performed by independent personnel employing independent test tools and techniques. IV&V has proven its worth on software development projects. Table 8-10 presents examples of the types of errors detected during various verification activities.

TABLE 8-10. EXAMPLES OF ERRORS DETECTED BY EACH VERIFICATION ACTIVITY (Reference 86)

<u>Verification Activity</u>	<u>Error Type</u>	<u>Error Cause</u>	<u>Error Effect</u>	<u>Detection Method</u>
Requirements analysis	Incomplete requirement	Partial specification of filter performance requirement	Unacceptable filter initialization/reinitialization	Comparison to existing system
	Missing requirement	Omitting error detection and recovery requirements	Insufficient program protection	Documentation analysis
	Incorrect requirement	Restart requirement cannot be met with existing hardware	Inadequate restart capability	Timing analysis
	Improper requirement allocation	Specifying accuracy for later processing within initialization requirements	Requirement overlooked during design	Independent derivation
	Untestable requirement	Requiring new program to meet or exceed accuracy of old program	Imprecise test: too many test cases; can't tell which is "better" if answers differ	Documentation analysis
Design analysis	Inefficient algorithm	Incorrect weighting factors for initializing real-time estimation algorithm	Output data remain "noisy" for twice as long as expected	Independent derivation/simulation
	Inadequate algorithm	Editing algorithm that detects only single-point failures	Biased data cause system failure	Independent derivation
	Incorrect algorithm	Incorrect criteria for selecting best data source	Poor-quality source selected, good source ignored	Logic analysis/simulation
	Logical error	Skipping logic to save data when current data contain errors	Misleading display of old data	Logic analysis
	Design oversight	No legality check of operator console commands before processing	Illegal command causes system crash	Comparison to references
Code analysis	Data accessing	Incorrect array index calculation for second, third, and fourth sensors	Meaningless output for second, third, and fourth sensors	Data structure analysis
	Sequencing error	Order of calculations inverted: using data before they are updated	Predicted value remains one cycle behind	Equation reconstruction
	Branching and iteration	Branch predicate required two contradictory conditions to be true	Unexecutable code	Logic reconstruction
	Interruptibility	Failure to save/restore two of three index registers	Meaningless results if higher priority interrupt occurs	Real-time conflict analysis
	Hardware interface	Overflow sets negative overflow to maximum positive value and sets positive overflow to maximum negative value	Misleading erratic jumps in real-time plots	Manual code analysis
Testing	Programming error	Negatively scaled constants converted to incorrect binary value; diagnostic routine made similar error in converting back to decimal	Inaccurate site constants that operator could not diagnose	Module test (dumped binary value)
	Missing capability	Hardware did not save data needed for power failure restart	Program could not recover from power failure	System test
	Insufficient accuracy	Double-precision divide and multiply accuracy could be as low as 6 digits (12 significant digits required)	Inaccurate program results	Module test
	Design oversight	Catastrophic restart destroyed time value	Loss of real-time velocity vs. time plot	System test

4. Automatic test analyzers should be used in verification/testing when appropriate. Although most automatic methods can be performed manually, additional errors can be inserted. Also, there is a lack of real-time automatic test analyzers. Development in this area would be a great benefit in verifying/testing digital avionic software.

8.5 HARDWARE VERIFICATION/TESTING.

8.5.1 BUS TESTING.

Overview. With the advent of digital avionics, more and more processing capability is being implemented at the local or subsystem level creating a need for highly reliable information flow between subsystems. Serial bus data transfer schemes offer increased reliability and reduced weight, volume, cost, and maintenance versus parallel bus or point-to-point information flow. From more than a dozen different serial data bus standards proposed over the last number of years, two standards have evolved into prominence: The ARINC 429 Digital Information Transfer System (DITS), and MIL-STD-1553B.

Little or no documented test procedures exist for either the ARINC 429 or MIL-STD-1553B. The Air Force has an internal verification facility for testing of MIL-STD-1553B subsystem called the System Engineering Avionics Facility (SEAFAC). They have developed a "Test Plan/Report for MIL-STD-1553" (reference 103) which currently only deals with remote terminals, one of the three possible generic types of subsystems available to be connected on a MIL-STD-1553B system. The test engineer is therefore forced to derive tests from the most up to date published standards for ARINC 429 and MIL-STD-1553B (references 104 and 105). However, the ARINC 429 specification is like all such other ARINC documents in that it is a form, fit, and function standard. These standards alone do not ensure satisfactory performance. Therefore, each use of ARINC 429 must be evaluated and certified individually using the best available means including experience and good judgment (reference 106).

Commercially available serial bus testers exist for both ARINC 429 and MIL-STD-1553B with the latter dominating the market. They perform functions in all or any combination of the three major areas of testing, monitoring, and simulating all or parts of a respective system. Bus testers provide testing of electrical signal parameters such as amplitude, bit rate, and slew rate, and also can detect various errors in the protocol. Bus monitors perform real-time data collection of bus traffic to be analyzed concurrently or at a later date. Several flight test MIL-STD-1553B monitors have been used including the Airborne Test Instrumentation System (ATIS) (reference 107), and the Digital Data Acquisition System (D-DAS) (reference 108). Bus simulators allow testing of target subsystems while simulating missing or unavailable subsystems and provides for error injecting in order to observe system responses. Overall, bus testers are a definite improvement over manual testing and should be used in conjunction with a detailed test plan for serial bus system qualification.

Data bus verification concerns center around having an extremely low probability of error in a data word transmission. Usual considerations such as voltage and temperature are the first level concerns. Parameters such as waveform asymmetry, signal rise time, wave shape distortion and wave shape offset must all be examined in order to assure that the system will operate at a given reliability or word transmission error rate. Table 8-11 shows the adverse impact on the word error

rate relative to variation in parameters mentioned above as experienced on the Space Shuttle Program which used a unique serial data bus transfer scheme having many of the same electrical characteristics as ARINC 429 and/or MIL-STD-1553B. Obviously, other serial data bus schemes must be analyzed separately, but table 8-11 does present a sense of the relative effects on reliability that generated by particular parameters have for that bus architecture (reference 93).

TABLE 8-11. PARAMETER VARIATION VERSUS IMPACT ON WORD ERROR RATE EXPERIENCED ON SPACE SHUTTLE PROGRAM

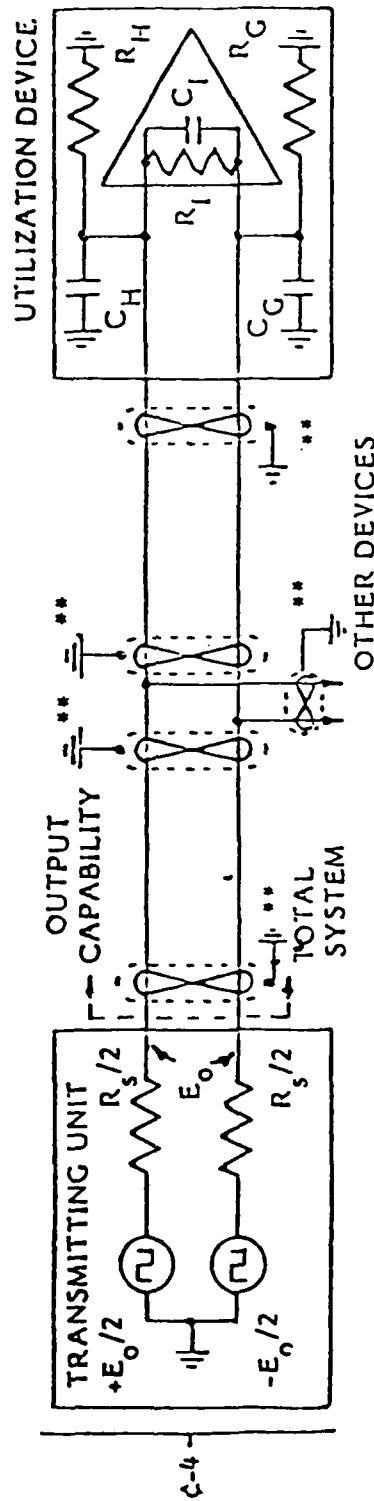
<u>Parameter Variation</u>	<u>Impact on Word Error Rate</u>
. Bit Asymmetry (@ 5%)	- Factor of 10
. Threshold Voltage	- Factor of 5
. Filter Bandwidth	- Factor of 5
. Power Supply Variation	- Affects threshold voltage and filter bandwidth
. Input Rise Time	- A factor of 2 around nominal value of 150 nanosecond
. Wave Shape Distortion	- Factor of 100 to 1000 for extreme distortion
. DC Offset	- At 150 mv noise, 2.6 V signal, a 25 mv DC offset results in a factor of 5 degradation
. Temperature	- Factor of 10 from -30 C to +92 C

8.5.2 ARINC 429-6.

Table 8-12 lists some important characteristics of the ARINC 429 serial bus standard. Additional detailed information on the different characteristics is presented below. The reader is directed to review the actual ARINC specification for clarification of or further details on the specifics of ARINC 429 (reference 88).

ARINC 429-6 is a serial bus. Data are transmitted over a shielded twisted pair cable as shown in figure 8-4. Notice that the shields should be grounded to the aircraft at both ends of all breaks. Figure 8-4 also shows the impedance and capacitance characteristics of interfaces connected to the ARINC bus.

A bipolar return to zero (BRZ) modulation scheme is used to send data as shown in figure 8-5. An HI signal represents a logic "one" and a LO signal represents a logic "zero". The rise and fall times are specified in order to minimize electromagnetic interference (EMI) radiated by the transmission line cable that might adversely affect other electronic equipment.



OUTPUT (SYSTEM) CAPABILITY

C-4 { Total System *Resistance

Total System *Capacitance

System Capacitance Unbalance

Bus Characteristic Impedance

UTILIZATION DEVICE STANDARDS

$R_i \geq 12,000 \text{ ohms}$

$C_i \leq 50 \text{ pF}$

$R_H \text{ or } R_G \geq 12,000 \text{ ohms}$
 $C_H \text{ and } C_G \leq 50 \text{ pF}$

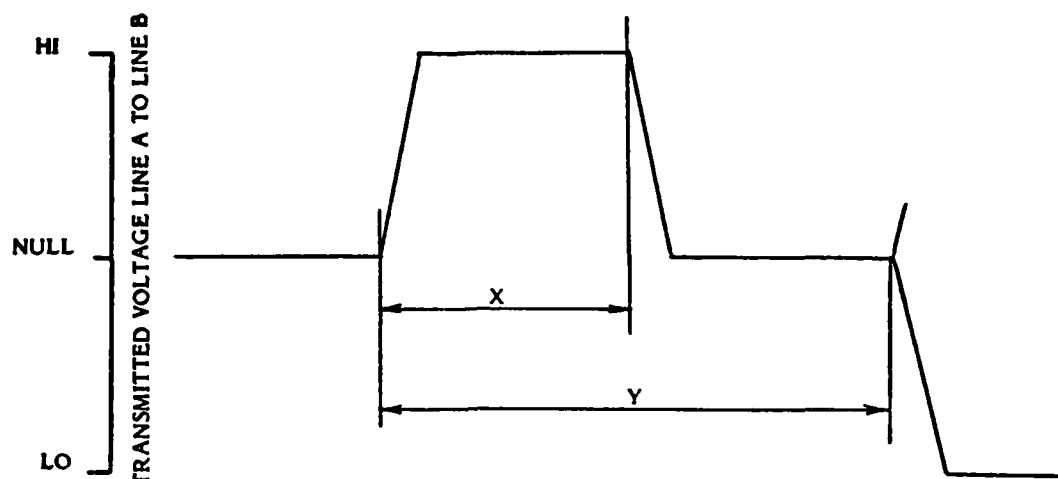
$R_S = 75 \text{ ohms}$

The total differential input impedance of the receiver should be limited to the values specified in Section 2.2.4.2.

NOTES

- * Includes aircraft interwiring
- ** Shields to be grounded in aircraft at both ends of all "breaks."

FIGURE 8-4. 429 INPUT/OUTPUT CIRCUIT STANDARDS



PARAMETER	HIGH SPEED OPERATION	LOW SPEED OPERATION
Bit Rate	100KBPS $\pm 1\%$	12 - 14.5KBPS
Time Y	10 usec $\pm 2.5\%$	Z* usec $\pm 2.5\%$
Time X	5 usec $\pm 5\%$	Y/2 $\pm 5\%$
Pulse Rise Time**	1.5 \pm 0.5 usec	10 \pm 5 usec
Pulse Fall Time**	1.5 \pm 0.5 usec	10 \pm 5 usec

* $Z = \frac{1}{R}$ where R = bit rate selected from 12 - 14.5KBPS range

** Pulse rise and fall times are measured between the 10% and 90% voltage amplitude points on the leading and trailing edges of the pulse and include permitted time skew between the transmitter output voltages A-to-ground and B-to-ground.

FIGURE 8-5. 429 OUTPUT SIGNAL TIMING TOLERANCES

TABLE 8-12. CHARACTERISTICS OF ARINC 429-6

Single twisted pair of wires

SIMPLEX: 1 transmitter and up to 20 receivers per bus

Bipolar Return to Zero (BRZ) modulation

32 bit word

FIXED FORMAT

Least Significant Bit (LSB) transmitted first

4 bit time gap between words

BROADCAST MODE OF TRANSMISSION (with a Command/Response mode option)

Low Bit Rate - 12.0 to 14.5 KHz $\pm 1\%$

High Bit Rate - 100 KHz $\pm 1\%$

Transmitter Voltage (See figure 8-6)

ERROR Detection

Parity (odd)

DATA Reasonableness

No requirement for stubbing, usually direct coupled

Defined DATA UP-DATE Rates per type of equipment

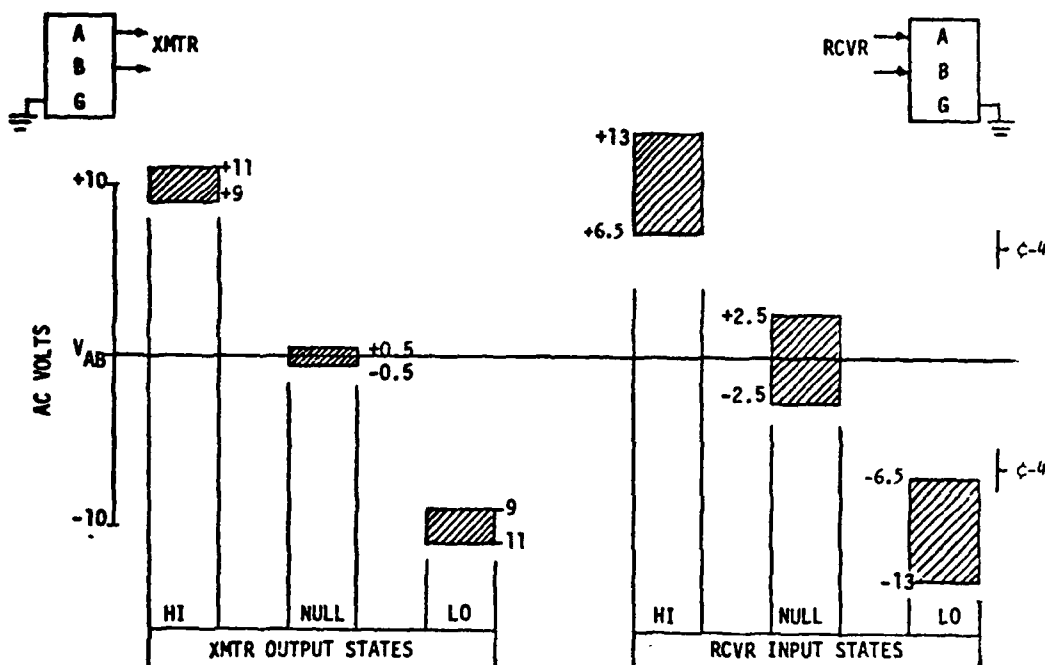


FIGURE 8-6. ARINC 429 BUS VOLTAGES

A 32-bit digital word is the basic information unit which is shown in figure 8-7. Each word consists of 1) an 8 bit label to define the type of data (e.g., ILS Frequency, DME distance); 2) a source/destination identifier (SDI) specifies 1 of 4 sources/destinations for words; 3) data; 4) a sign/ status matrix indicates the sign (plus, minus, east, west, etc.) of the data and the status of the transmitter hardware; and 5) a parity bit (odd). The bits are numbered in order of transmission; i.e., bit 1 is transmitted first and bit 32 is transmitted last.

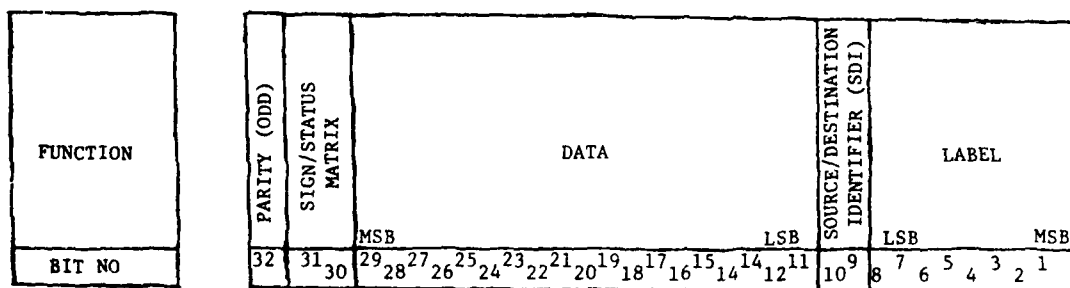


FIGURE 8-7. BASIC ARINC 429 WORD

The ARINC 429 data words are predefined per the 8-bit label code. Depending upon this label code, the remaining bits may maintain the same categories as in the basic word or they may take on new meanings. There are five application groups defined by the label. Refer to the ARINC 429 specification (reference 88) for further details.

A broadcast transmission protocol is a method where the single transmitter sends words out over the bus and all receivers connected to that bus listen to the transmission. If multiple words are to be sent, the transmitter must wait a minimum of 4 bit times before sending the next word. The receiver examines the label code (first 8 bits) to determine if it will accept or reject the rest of the word. The transmitter has no feedback as to whether the receiver correctly acquired the word. There is a command/response mode option which involves sending status information over an auxiliary bus connected between the two subsystems (see the ARINC 429 specification for details).

The high speed operation is defined as 100 kilobits per second with a tolerance of ± 1 percent and the low speed operation is defined as 10.0 to 14.5 kilobits per second also within ± 1 percent. However, there is an advisory note stating "That designers should avoid the selection of 13.6 kilobits per second for the low-speed operations and precisely 100 kilobits per second for high-speed operation to ensure that the system is not responsible for interference to OMEGA and LORAN C systems with which the aircraft might be equipped." (Reference 88)

The following is an excerpt from the ARINC 429-6 specification. "Air transport industry experience with digital information transfer systems predating the Mark 33 DITS has shown that the twisted shielded pair of wires can be regarded as high integrity link unlikely to introduce bit errors into the data passing through it. It is for this reason that no means for error correction are specified in this document. The error detection capability specified above may be used as desired in receiving terminals. Also, the data may be submitted to reasonableness checks. Data intended for human consumption in the cockpit are normally smoothed before transmission to ensure tolerable levels of display jitter. As this process eliminates any obviously wild data points, the need for further error detection is questionable." (Reference 88)

ARINC Specification 429-6 does not specify the required performance of the data bus, but only general characteristics. For example, bit error rate requirements are omitted, cable parameters, except characteristic impedance (75 ohms), are not controlled, and no formal requirement is placed on the number or length of stubs (reference 90). Stubbing is the method whereby a separate line is connected between the primary data bus line and a subsystem as shown in Figure 8-8.

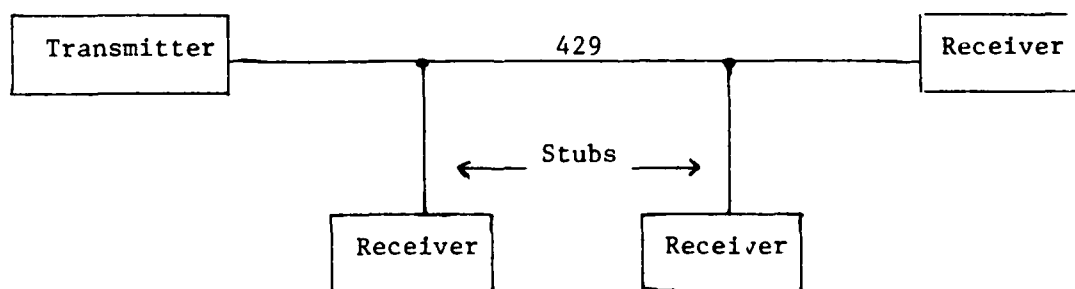


FIGURE 8-8. STUBS IN A 429 BUS SYSTEM

The direct connection of a stub line causes an impedance mismatch which appears in the waveforms. This mismatch can be reduced by filtering at the receiver. The preferred method of stubbing is to use transformer coupled stubs. This method provides the benefits of DC isolation, increased common mode protection, a doubling of the effective stub impedance, and fault isolation for the entire stub and connected subsystem. Direct coupled stubs provide no DC isolation or common mode rejection (reference 89).

The transmitting subsystem in an ARINC 429 system is to supply data at sufficiently high rates to ensure that only small incremental value changes occur between updates. For example, the specification states that all radio systems' nominal update rate should be five times per second (every 200 msec).

8.5.3 MIL-STD-1553B.

Both versions, A and B, are presented here since both versions are used in current aircraft and aircraft developments. The following information carefully distinguishes between the two versions where applicable.

Table 8-13 provides detail on the physical and electrical data bus characteristics (reference 94).

MIL-STD-1553B is a bidirectional serial data bus supporting up to 32 remote terminals per bus or buses. The bipolar non return to zero (BNRZ) modulation scheme, Manchester II is utilized as shown in figure 8-9 (reference 89).

The three types of transmission words available for MIL-STD-1553B are depicted in figure 8-10. Notice that the command and status words have a fixed format while the data word is in free format. The words are distinguished by a 3-bit time sync pattern. The distinction between command and status, which have the same sync pattern, is discussed below under the command/response protocol area (reference 89). The order of transmission is that bit time #1 or the beginning of the sync is transmitted first.

The bus controller shall provide a minimum gap time of 4.0 microseconds between messages shown as T on figure 8-11. The remote terminal shall respond to a valid command word within 4.0 to 12.0 microseconds also shown as T in figure 8-11 (Reference 89).

MIL-STD-1553B operates with a command/response protocol with a broadcast mode option. Figure 8-12 shows the different transfers that are possible over the bus. The first shown are the main protocol types (reference 89).

Table 8-14 describes the output voltage levels, waveform, acceptable noise, symmetry with direct and transformer coupling (reference 94).

The United States Air Force released Notice 1 concerning MIL-STD-1553B on February 12, 1980, which updates the specification (see Reference 95). The following is an abbreviated list of the changes put forth applicable to all U.S. Air Force aircraft internal avionics activities.

TABLE 8-13. COMPARISON OF DATA BUS CHARACTERISTICS

Requirement	Requirement/Paragraph/Figure	
	MIL-STD-1553B	MIL-STD-1553A
1. Twisted, Shielded, Jacketed	yes 4.5.1.1	yes 4.2.4.1
2. Minimum Cable Shield Coverage	75% 4.5.1.1	80% 4.2.4.1
3. Minimum Cable Twist	4 twist/ft 4.5.1.1	1 twist/in. (12 twist/ft) 4.2.4.1
4. Wire-to-Wire Distributed Capacitance (Maximum)	30 pF/ft 4.5.1.1	30 pF/ft 4.2.4.1
5. Characteristic Impedance of Cable	Nominal 70 - 80 at 1 MHz 4.5.1.2	70 \pm 10% at 1 MHz 4.2.4.2
6. Cable Attenuation	1.5 dB/100 ft at 1 MHz 4.5.1.3	1 dB/100 ft at 1 MHz 4.2.4.3
7. Cable Length	unspecified	100 ft maximum 4.2.4.4
8. Cable Termination Using a Resistance at Both Ends	Nominal Characteristic Impedance \pm 2% 4.5.1.4	Characteristics Impedance 4.2.4.6
9. Cable Stubbing	Transformer coupling of direct coupling stub length maximum 20 ft unless installation dictates longer stubs 4.5.1.5.1 or 4.5.1.5.2 Figure 9 or Figure 10	Transformer coupling for stubs greater than 1 ft and less than 20 ft. Direct coupling less than 1 ft maximum stub 20 ft 4.2.4.5 Figure 7
10. Cable Coupling (Connector)	unspecified	Compatible with Amphenol type 31-235 or Trompeter type TEI-14949-El37 receptacles and Amphenol type 31-224 or Trompeter type TEI-14949-El36 plugs 4.2.4.6

TABLE 8-13. COMPARISON OF DATA BUS CHARACTERISTICS (Continued)

Requirement	Requirement/Paragraph/Figure MIL-STD-1553B	Requirement/Paragraph/Figure MIL-STD-1553A
11. Cable Coupling Shielded Box	75% coverage minimum 4.5.1.5.1.3 and 4.5.1.5.2.2	Shielded coupler box 4.2.4.6
12. Coupling Transformer Turns Ratio	1:1.41 \pm 3% higher turns on isolation resistor side of stub 4.5.1.5.1.1	unspecified --
13. Transformer Open Circuit Impedance	3000 ohms over frequency of 75 KHz -1 Hz with 1 v rms sine wave 4.5.1.5.1.1.1	unspecified --
14. Transformer Waveform Integrity	Droop not to exceed 20% overshoot and ringing less than \pm 1 v peak undertest of figure 11 4.5.1.5.1.1.2	unspecified --
15. Transformer Common Mode Rejection	45 dB at KHz 4.5.1.5.1.1.3	unspecified --
16. Fault Isolation isolation resistor in series with data bus cable (coupler) For direct coupled case with the isolation resistor in the RT	$R = 0.75 Z_0 \pm 2\%$ 4.5.1.5.1.2 -- $R = 55 \text{ ohms} \pm 2\%$ 4.5.1.5.2.1 -- Figure 10	$R = 0.75 Z_0 \pm 5\%$ 4.2.5.2 Figure 7
17. Impedance across the data bus for any failure of coupling transformer, cable stub or terminal receiver/ transmitter transformer coupling Direct Coupling	No less than 1.5 Z_0 Z_0 = cable nominal characteristic impedance 4.5.1.5.1.2 No less than 110 ohms 4.5.1.5.2.3	No less than 1.5 Z_0 Z_0 = cable impedance 4.2.5.2
18. Stub Voltage Requirements/Input Level Transformer Coupling	**Range of 1.0 - 14 vp-p line-to-line with one fault as stated in 17 above 4.5.1.5.1.4 Figure 9	**Range of the 0.5 - 10 v p 1.0 - 20 v p-p line-to-line 4.2.5.4.1 Figure 7
Direct Coupling	**Range of 1.4 - 20 vp-p line-to-line with one fault as stated in 17 above 4.5.1.5.2.3 Figure 10	**Range of the 0.5 - 10 v p 1.0 - 20 v p-p line-to-line 4.2.5.4.1 Figure 7
** Assumes one fault of a coupling transformer, cable stub or terminal receiver/transmitter		
19. Wiring and Cabling for EMC	MIL-E-6051 4.5.1.5.3 MIL-STD-1553B	MIL-E-6051 4.2.4.7 MIL-STD-1553A

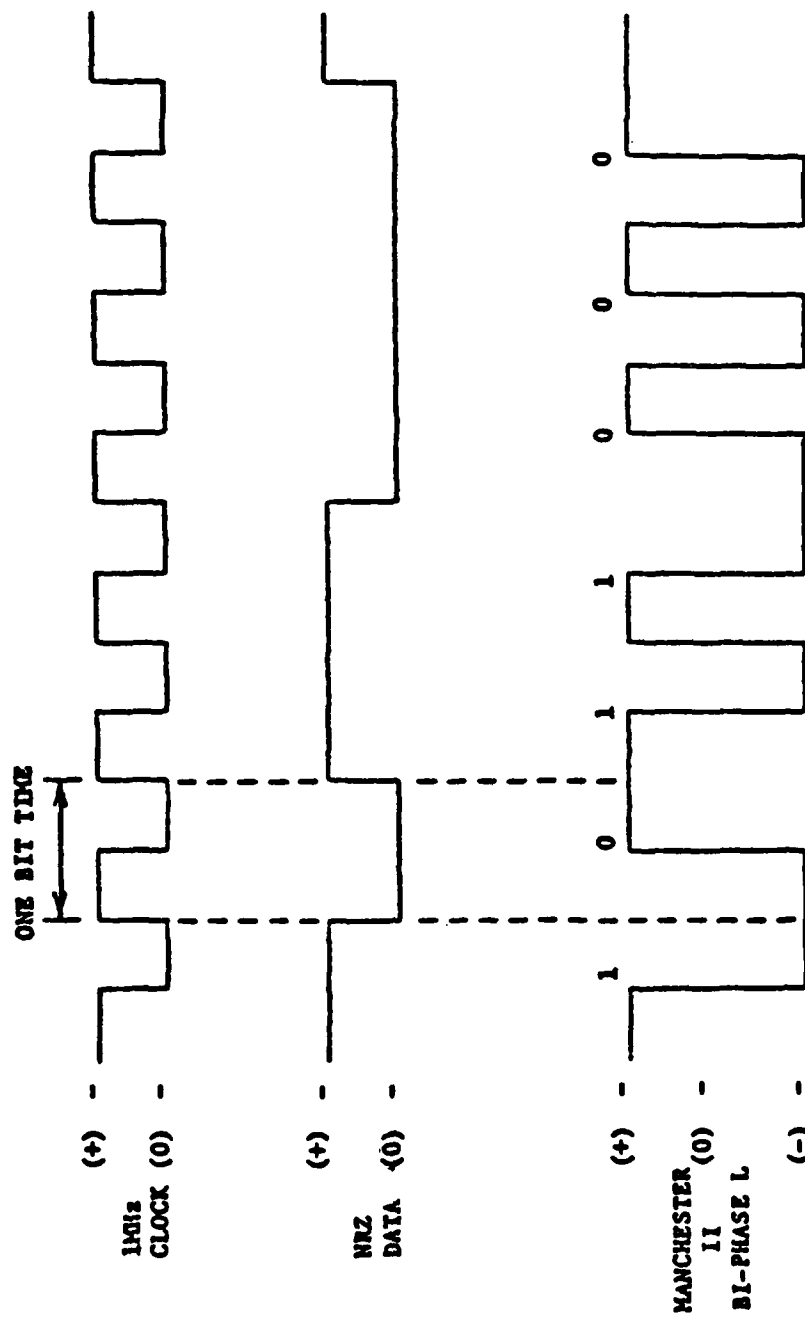


FIGURE 8-9. DATA ENCODING MANCHESTER II

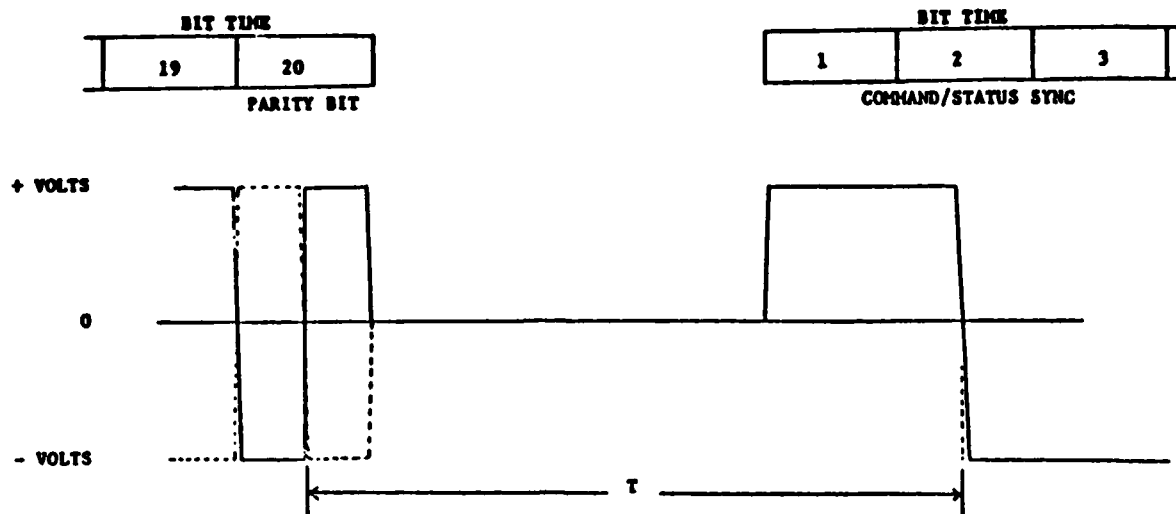
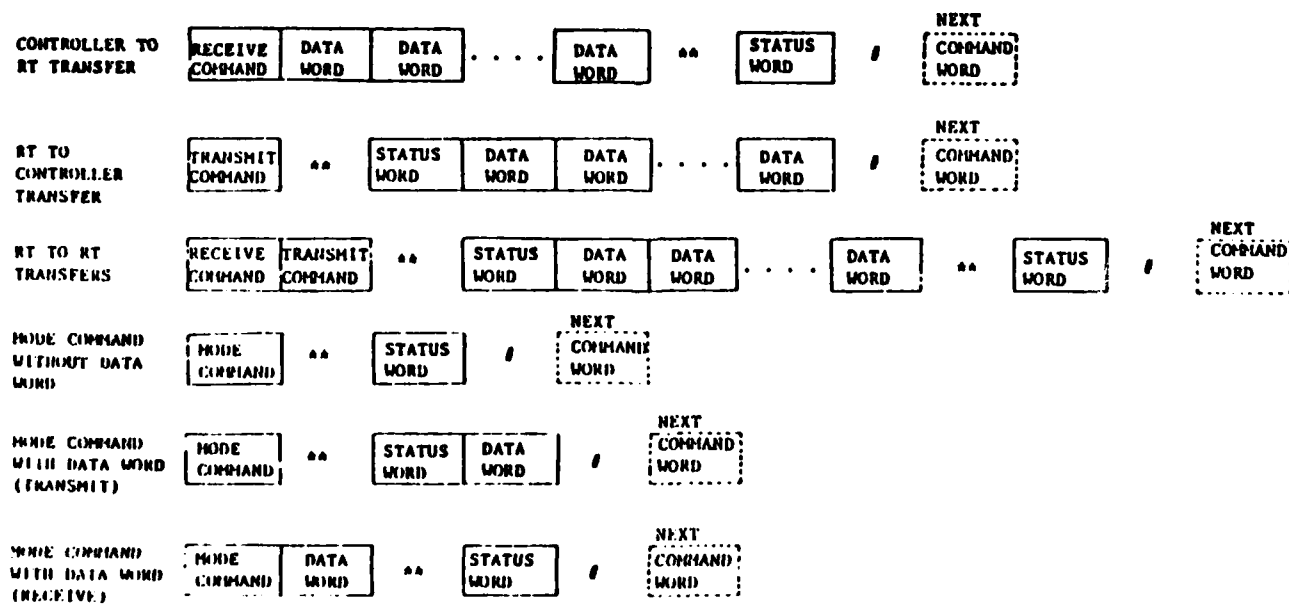


FIGURE 8-11. INTERMESSAGE GAP AND RESPONSE TIME



NOTE: / INTERMESSAGE GAP
** RESPONSE TIME

FIGURE 8-12. INFORMATION TRANSFER FORMATS

TABLE 8-14. COMPARISON OF TERMINAL CHARACTERISTICS

Requirement	Requirement/Paragraph MIL-STD-1553B	Requirement/Paragraph MIL-STD-1553A
1. Output Level Transformer Coupling	With $R_L = 70 \pm 2X$ 18.0 - 27.0 vp-p line-to-line 4.5.2.1.1.1 Figure 12	+3.0 - +10 v peak (6.0 - 20.0 vp-p) line-to-line with no faults with one fault of a coupling transformer, cable stub, or terminal receiver/transmitter +2.25 - +11.25 v peak (4.5 - 15 v p-p) line-to-line 4.2.5.3.1
Direct Coupling	with $R_L = 35 \pm 2X$ 6.0 - 9.0 vp-p line-to-line 4.5.2.2.1.1 Figure 12	
2. Output Waveform Zero Crossing Deviation	+ 25 ns 4.5.2.1.1.2 Figure 12	+ 25 ns 4.2.5.3.2 Point C, Figure 7 and Figure 8
Rise and Fall Time (10% - 90%)	100 - 300 ns 4.5.2.1.1.2 Figure 13	\geq 100 ns 4.2.5.3.2 Figure 8
Transformer Coupling Distortion (Including overshoot and ringing)	+ 900 mv peak line-to-line 4.5.2.1.1.2 Point A, Figure 12	unspecified
Direct Coupling Distortion (Including overshoot and ringing)	+ 300 mv peak line-to-line 4.5.2.2.1.2 Point A, Figure 12	unspecified
3. Output Noise Transformer Coupling	14 mv RMS line-to-line 4.5.2.1.1.3 Point A, Figure 12	10 mv p-p line-to-line 4.2.5.3.3 Point A, Figure 7
Direct Coupling	5 mv RMS line-to-line 4.5.2.2.1.3 Point A, Figure 12	
4. Output Symmetry (after 2.5 ns of mid-bit crossing of the last parity bit Transformer Coupling)	+ 250 mv peak line-to-line 4.5.2.1.1.4 Point A, Figure 12	unspecified
Direct Coupling	+ 90 mv peak line-to-line 4.5.2.2.1.4 Point A, Figure 12	unspecified
5. Input Waveform Maximum Zero Crossing Deviation	+ 150 ns 4.5.2.1.2.1 Point A, Figure 9 or Figure 10	unspecified
Input Signal Response Range Transformer Coupling	0.06 - 14.0 vp-p line-to-line 4.5.2.1.2.1 Point A, Figure 9	+ 0.5 - + 10.0 v peak (1.0 - 20 vp-p) line-to-line 4.2.5.4.1 Point C, Figure 7

TABLE 8-14. COMPARISON OF TERMINAL CHARACTERISTICS (Continued)

Requirement	Requirement/Paragraph MIL-STD-1553B	Requirement/Paragraph MIL-STD-1553A
Direct Coupling	1.2 - 20 vp-p line-to-line 4.5.2.2.2.1 Point A, Figure 10	
Input Signal No Response Range Transformer Coupling	0.0 - 0.2 vp-p line-to-line 4.5.2.1.2.1 Point A, Figure 9	unspecified
Direct Coupling	0.0 - 0.28 vp-p line-to-line 4.5.2.2.2.1 Point A, Figure 10	unspecified
6. Common Mode Rejection	± 10 v peak line-to-ground DC to 2 MHz 4.5.2.1.2.2 or 4.5.2.2.2.2 Point A, Figure 9 or Figure 10	± 10 v peak line-to-ground DC to 2 MHz 4.2.5.4.2 Point A, Figure 7
7. Input Impedance Transformer Coupling	Minimum of 1000 ohms over a frequency range of 75 KHz - 1 MHz line-to-line 4.5.2.1.2.3 Point A, Figure 9	Minimum of 2000 ohms over a frequency range of 100 KHz - 1 MHz line-to-line 4.2.5.4.3 Point C, Figure 7
Direct Coupling	Minimum of 2000 ohms over a frequency range of 75 KHz - 1 MHz line-to-line 4.5.2.2.2.3 Point A, Figure 10	
8. Noise Rejection/Error Rate Transformer Coupling	Maximum of one part in 10^7 word error in the presence of additive white Gaussian noise of 140 mv RMS over a bandwidth of 1.0 KHz - 4 MHz. Input voltage 2.1 vp-p line-to-line Point A, figure 9 accept/reject table II 4.5.2.1.2.4	Maximum bit error rate of 10^{-12} and a maximum incomplete message rate of 10^{-6} . In a configuration of one bus controller on a 20 ft stub with a minimum of 100 ft of main bus cable between coupling boxes. The test is conducted in presence of magnetic field per MIL-STD-462 method RS02 (spike test) with the limits of MIL-STD-461 RS02. 4.3.3
Direct Coupling	Maximum of one part in 10^7 word error rate in the presence of additive white Gaussian noise of 200 mv RMS over a bandwidth of 1.0 KHz - 4 MHz. Input voltage 3.0 vp-p line-to-line Point A, figure 10 accept/reject table II 4.5.2.2.2.4	

- (1) Will not be supported
 - Dynamic Bus Control
 - Broadcast Command
 - Direct Coupled Stubs
- (2) Must be supported
 - Dual Standby Redundant BUSES

8.6 SYSTEM INTEGRATION/VERIFICATION/VALIDATION.

The flight test environment provides the only truly credible qualification results for avionic systems. However, cost and limited visibility into the system operation are serious drawbacks (reference 96). Based on the L-1011 flight test program during 1976, the cost of a flight test hour was estimated at \$10,000 per hour (reference 97). The projected number of flight test hours for certification of the Boeing 767 is 1200 hours (reference 98). This would make the cost of flight test to be a very conservative estimate of 12 million dollars. System monitoring is limited due to restricted instrumentation capabilities, space constraints, and flyable test equipment.

The above drawbacks to flight test have resulted in use of various simulation activities during system integration, verification, and validation of both hardware and software. Data obtained from the simulations used during the various development phases are then correlated with results obtained from flight test. This correlation of results is used to lend confidence to the results from the simulations and therefore allowing the simulation results to be used as evidence in system validation and certification. Increased use of simulation reduces the total number of flight test hours and produces more detailed information about the internal operations of the system.

The simulation categories of interest to avionic system manufacturers are: Ground, hot bench, iron bird, and airborne simulations. These simulation practices and a discussion of system validation through the use of flight test are described below.

8.6.1 Ground Simulation.

Ground simulation is the least complex of the different simulation categories used in system verification and validation. Typically, it includes only one host computer used to emulate the target hardware or used to verify by simulation the control laws and algorithms to be implemented in software.

Emulation involves programming the host computer so that it "looks" like the target hardware. The software to be embedded into the target hardware can then be run on the host computer.

Simulation of control laws and algorithms is used to verify that the control laws can be feasibly implemented in software and helps identify critical software modules. These critical modules will then receive additional attention during the development and testing phases.

The above activities provide verification of system software versus software requirements, and sets the baseline for control laws against which future development stages can be verified. System validation, as defined in this handbook, is not accomplished using only ground simulation as described above.

8.6.2 Hot Bench

A hot bench is a facility where LRU's and associated embedded software are tested to see if the requirements are met and subsequently modified if necessary to meet these requirements. The hot bench is a complex combination of hardware and software with a number of aids available for use during checkout.

Figure 8-13 shows the typical elements that make up a hot bench facility. The LRU under test may be coupled with other LRU's to form a system configuration to be tested. The test control and monitor panel provide debugging aids as discussed below. Environmental simulators provide the necessary external stimuli to drive the system through its test procedures (reference 99).

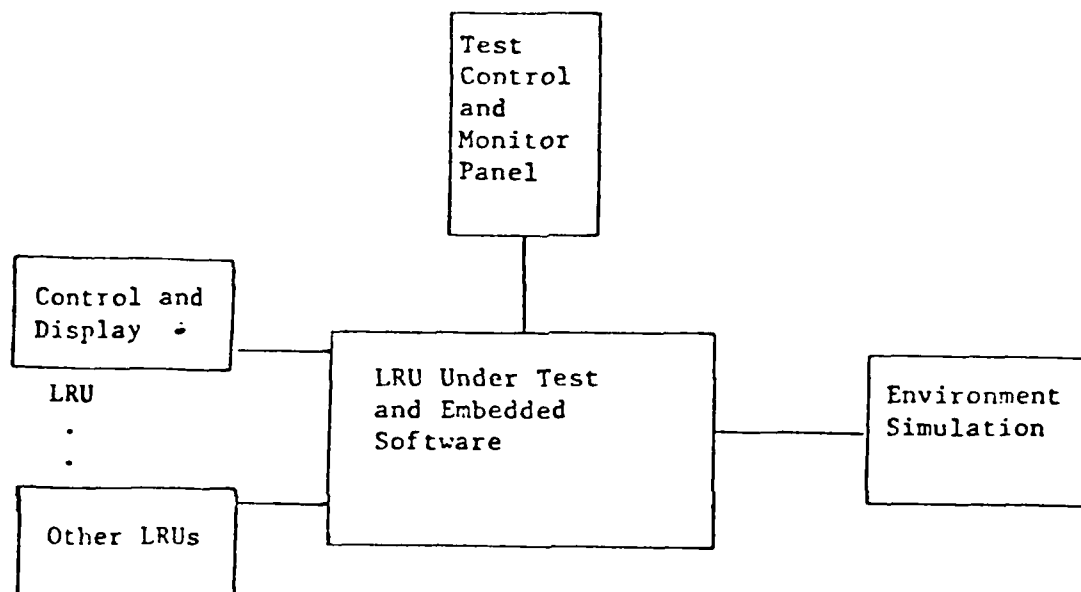


FIGURE 8-13. HOT BENCH FACILITY

Debugging aids of the hot bench center around software monitor capabilities. Typically they include the following functions:

- (1) Breakpoints can be set where execution will stop
- (2) Execution can start at any location
- (3) Memory contents can be modified
- (4) Single step through instructions
- (5) Examine content of memory and registers (reference 99)

Subsystem hardware and software verification and validation can be performed on a hot bench system. The LRU and embedded software are exactly the same as the equipment and configuration used within an aircraft. Assuming that comprehensive testing occurs, as required in the LRU test plan, validation of the LRU against its subsystem requirements can be achieved. At the very least, the results of hot bench testing can be used to add support to the results of higher level simulation or flight test.

8.6.3 Iron Bird.

Current validation practices rely heavily on the use of Iron-Bird facilities, such as that shown in figure 8-14. Iron Bird is the most complex simulation involving multiple computers operating in a closed-loop real time environment. It is a natural extension of hot bench simulation but now as much of the system as possible is implemented using the actual avionics and airframe subsystems. Note that because an iron bird is so close to being an aircraft, it costs approximately the same to maintain the simulator and actual aircraft. The operating costs are much lower per simulator hour (1/10 to 1/4) than per aircraft flight hour.

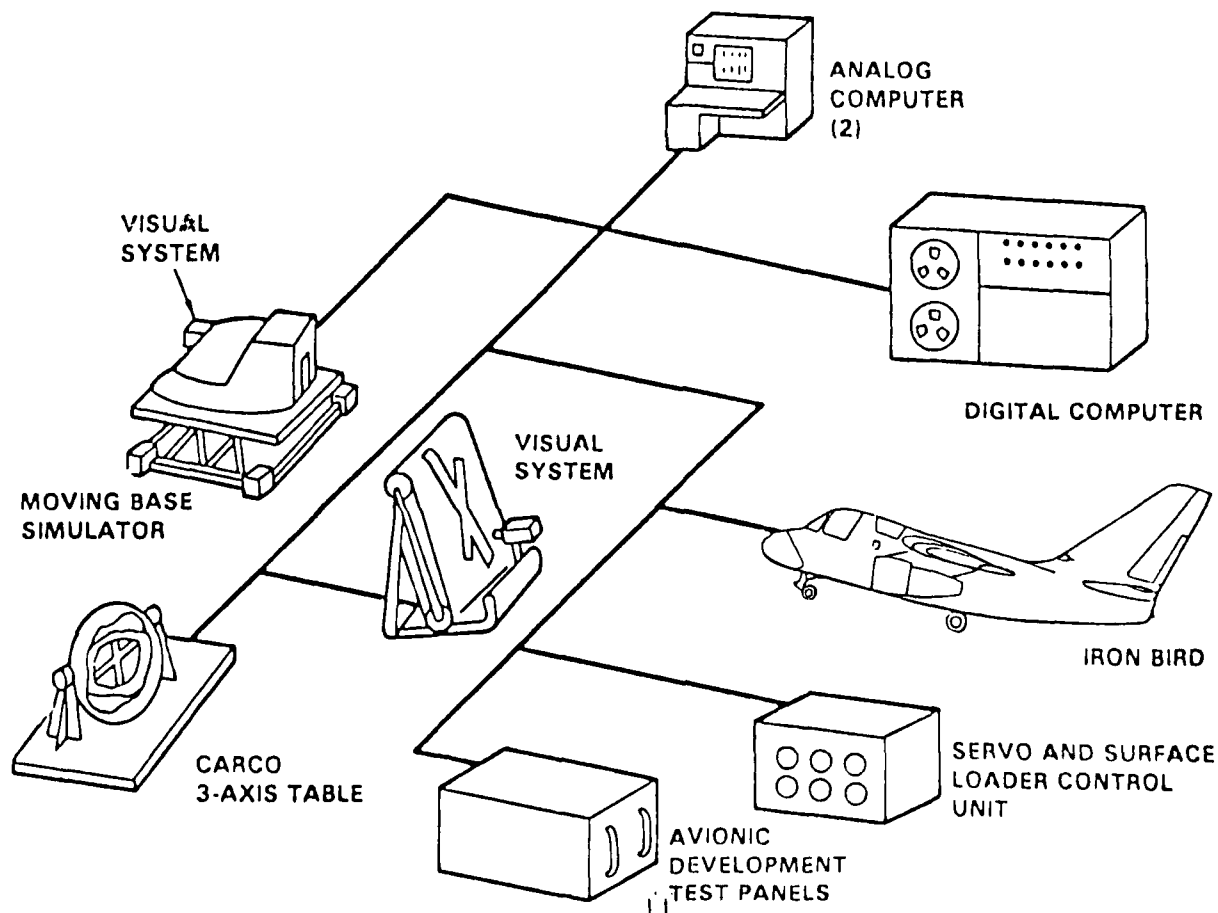


FIGURE 8-14. IRON BIRD OVERVIEW

Normal and faulted operations are tested on the iron bird facility. Also, safety and failure modes that are too dangerous to attempt to test during flight test are conducted on the iron bird.

Generally, the iron bird is where the major portion of system validation occurs. Results from ground simulation and hot bench are verified against the iron bird results.

8.6.4 Airborne Simulation.

When a particular subsystem is unavailable for testing of a related subsystem with which it must interact, the unavailable system can be simulated. This permits the completion of system validation of the target subsystem in an orderly time frame. The simulation equipment must be able to operate in an aircraft if the system validation is being performed in flight test. Also, this is a means of certifying equipment for numerous different aircraft while actually only flight testing on one aircraft.

8.6.5 Flight Test.

As stated previously, flight test is the final step in validation. Flight test experiments are a well chosen subset of the tests performed during iron bird system validation. By doing this, a correlation of the iron bird simulation results can be made with the flight test results thereby validating all of the iron bird results.

8.7 REFERENCES.

1. Waterman, Hugh E., FAA's Certification Position on Advanced Avionics, Astronautics and Aeronautics, May 1978, pp. 49-51.
2. Waterman, Hugh E., FAA Role of Certification: Advanced Aircraft Systems, AIAA Paper 77-1472, AIAA 2nd Digital Avionics Systems Conference, Los Angeles, November 2-4, 1977.
3. Ness, W. G., McCrary, W. C., Bridgman, M. S., Hitt, E. F., and Kenney, S. M., Automated Reliability and Failure Effects Methods for Digital Flight Control and Avionic Systems, NASA CR-166148, Lockheed-Georgia Company and Battelle-Columbus Laboratories, March 1981.
4. Thatcher, R. K., et al, Development of Unified Reliability Prediction Techniques for Flight Control Systems, Technical Report AFFDL-TR-70-81 (AD 877179), Battelle's Columbus Laboratories, for Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, June 1970.
5. Chelson, P. O., and R. E. Eckstein, Reliability Computation from Reliability Block Diagrams, Jet Propulsion Laboratory, Technical Report 32-1543, December 1971.
6. Carter, W. C., et al, Design Techniques for Modular Architecture for Reliable Computer Systems, IBM T. J. Watson Research Center, Report RA-12, NASA Report 70-208-0002, March 1970.
7. Fleming, J. L., Relcomp: A Computer Program for Calculating System Reliability and MTBF, IEEE-TR-20, No. 3, pp. 102-107, August 1971.
8. Ng, Y. W., Reliability Modeling and Analysis for Fault-Tolerant Computers, National Science Foundation Report No. NSF-MCS-7203633-76981; Ph.D. dissertation, University of California, Los Angeles, Report No. UCLA-ENG-7698, September 1976.
9. Ng, Y. W., and A. Avizienis, ARIES - An Automated Reliability Estimation System for Redundant Digital Structures, Proceedings of the 1977 Annual Reliability and Maintainability Symposium, Philadelphia, PA, January 1977, pp. 108-113.
10. Ng, Y. W., and A. Avizienis, A Reliability Model for Gracefully Degrading and Repairable Fault-Tolerant Systems, Proc. FTC-7, Los Angeles, 1977, pp. 22-28.
11. Ng, Y. W., and A. Avizienis, ARIES 76 User's Guide, National Science Foundation Report No. NSF-MCS-7203633-78944, University of California, Los Angeles Report No. UCLA-ENG-7894, December 1978.
12. Reliability Model Derivation of a Fault-Tolerant, Dual, Spare-Switching Digital Computer System, Prepared for NASA-Langley Research Center by Ratheon Company, Contract NAS1-12668, March 1974.

13. Stiffler, J. J., Computer-Aided Reliability Estimation, AIAA/NASA/ IEEE/ACM Computers in Aerospace Conference, Los Angeles, CA, 1977.
14. Bjurman, B. E., et al, Airborne Advanced Reconfigurable Computer System (ARCS), NASA-CR-145024, Prepared for Langley Research Center, NASA by Boeing Commercial Airplane Company under Contract NAS1-13654, August 1976.
15. Masreliez, Johan C., and Bo E. Bjurman, Fault-Tolerant System Reliability Modeling/Analysis, Journal of Aircraft, ol. 14, No. 8, pp. 803-808, August 1977.
16. Private Communications with Johan C. Masreliez, Honeywell, Marine Systems Division, Seattle, Washington.
17. Private Communications with Roger H. Edwards, Boeing Commercial Airplane Company, Seattle, Washington.
18. Private Communications with Bob Ebner, Litton Industries, Los Angeles, California.
19. Conn, R. B., et al, Definition and Tradeoff Study of Reconfigurable Airborne Digital Computer System Organizations, Prepared for NASA-Langley Research Center by Ultrasystems, Incorporated, Contract NAS1-12793, November 1974.
20. Conn, R. B., et al, CAST - A Complementary Analytic - Simulative Technique for Modeling Complex, Fault-Tolerant Computing Systems, AIAA Computers in Aerospace Conference, Los Angeles, California, November 1977.
21. Stiffler, J. J., et al, CARE III Final Report, Phase I, NASA CR159122, November 1979.
22. Mulcare, D. B., Rang, E. R., Comprehensive Methodologies for Digital Flight Control Software Verification, NAECON 1980, pp. 252-259.
23. Myers, G. T., Software Reliability, Wiley-Interscience Publication, 1976.
24. Adrion, W. R., Branstad, M. A., Cherniavsky, J. C., Validation, Verification, and Testing for the Individual Programmer, IEEE Computer, December 1980.
25. Adrion, W. R., Branstad, M. A., Cherniavsky, J. C., Validation, Verification, and Testing of Computer Software, NBS PB81-167074, February 1981.
26. Guidelines for Documentation of Computer Programs and Automated Data Systems, FIPS 38, Federal Information Processing Standards Publications, U.S. Department of Commerce/National Bureau of Standards, Washington, D. C., February 1976.
27. Buckley, F., A Standard for Software Quality Assurance Plans, IEE Computer, Vol. 12, No. 8, August 1979, pp. 43-50.
28. IEEE Draft Test Documentation Standard, IEEE Computer Society Technical Committee on Software Engineering, Subcommittee on Software Standards, New York (1979).

29. Mills, H. D., Top Down Programming in Large Systems, in Debugging Techniques in Large Systems, R. Rustin ed., Prentice-Hall, 1970, pp. 41-55.
30. Yourdon, E., and Constantine, L. L., Structured Design, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.
31. Jackson, M. A., Principles of Program Design, Academic Press, New York, 1975.
32. Myers, G. J., The Art of Software Testing, John Wiley and Sons, New York (1979).
33. Baker, F. T., Chief Programmer Team Management of Production Programming, IBM Systems Journal, Vol. 11, No. 1, pp. 56-73 (1972).
34. Fagan, M. E., Design and Code Inspections to Reduce Errors in Program Development, IBM Systems Journal, Vol. 15, No. 3, pp. 182-211 (1976).
35. Weinberg, G. M., The Psychology of Computer Programming, Van Nostrand Reinhold, New York, 1971.
36. Technical Reviews and Audits for Systems, Equipment, and Computer Programs, MIL-STD-1521A (USAF), U.S. Department of the Air Force, Washington, D. C. (1976)
37. Gentzen, G., Investigations into Logical Deductions, in The Collected Works of Gerhard Gentzen, M. E. Szabo ed., North-Holland, Amsterdam, pp. 68-128, 1969.
38. Church, A., Introduction to Mathematical Logic, Vol. 1, Princeton University Press, Princeton, 1956.
39. Ambler, A. L., Good, D. I., Browne, J. C., Burger, W. F., Cohen, R. M., Hoch, C. G., and Wells, R. E., Gypsy: A Language for Specification and Implementation of Verifiable Programs, Proceedings of an ACM Conference on Language Design for Reliable Software, D. B. Wortman ed., New York, pp. 1-10 (1978).
40. Robinson, L., The HDM Handbook, Volume I-III, SRI Project 4828, SRI International, Menlo Park (1979).
41. Manna, Z., Mathematical Theory of Computation, McGraw-Hill, New York, 1974.
42. Elspas, Levitt, K. N., Waldinger, R. J., and Waksman, An Assessment of Techniques for Proving Program Correctness, Computing Surveys, Vol. 4, No. 2, pp. 97-147 (1972).
43. Gerhart, S. L., etc., An Overview of AFFIRM: A Specification and Verification System, Proceedings IFIP Congress 1980, North-Holland Publishing Company, Amsterdam, to appear (1980).
44. Constable, R. L., and O'Donnell, M. J., A Programming Logic, Winthrop Publishing Company, Cambridge (1978).
45. Fairley, R. E., Tutorial: Static Analysis and Dynamic Testing of Computer Software, IEEE Computer, April 1978, pp. 14-23.

46. Reifer, D., and Trattner, S., A Glossary of Software Tools and Techniques, IEEE Computer, July 1977.
47. Stewart, M., C-26 Verification Tools, R:DS-722-2R1140, Logicon, San Pedro, CA, 1972.
48. Slavinski, R. T., Static FORTRAN Analyzer, RADC-TR-75-275, November 1975.
49. Hoffman, R. H., NASA/Johnson Space Center Approach to Automated Test Data Generation, Proceedings of Computer Science and Statistics: Eighth Annual Symposium on the Interface, available from UCLA, February 1975.
50. Shipple, L. K., and Pitts, M. A., A User's Appraisal of an Automated Program Verification Aid, AFAL-TR-75-242, December 1975.
51. Sammet, J. E., Programming Languages: History and Fundamentals, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1969.
52. Fisher, D. A., Programming Language Commonality in the Department of Defense, Defense Management Journal, October 1975, pp. 29-33.
53. Boehm, B. W., McClean, R. K., and Urgrig, D. B., Some Experience with Automated Aids to the Design of Large-Scale Reliable Software, Proceedings of the International Conference on Reliable Software, IEEE Catalog No. 75CH0940-7CSR, April 1975, pp. 105-113.
54. King, J. C., A New Approach to Program Testing, Proceedings of the International Conference on Reliable Software, IEEE Catalog No. 75CH0940-7 CSR, April 1975, pp. 105-113.
55. Elspas, B., etc., An Assessment of Techniques for Proving Program Correctness, Computing Surveys, June 1972, pp. 97-147.
56. Brown, J. R., and Hoffman, R. H., Automating Software Development: A Survey of Techniques and Automated Tools, TRW-SS-72-03, TRW Systems Group, Redondo Beach, CA, May 1972.
57. Carpenter, L. C., DECA, Design Expression and Confirmation Aid, Proceedings, Computer Science Conference, ACM, February 1974.
58. Lanzano, B. C., Program Automated Documentation Methods, TRW-SS-70-04, TRW Systems, Redondo Beach, CA, November 1970.
59. Poseidon MK 88 Fire Control System Computer Program Verification and Validation Techniques Study, Volume III, Ultrasystems, Incorporated, 500 Newport Center Drive, Newport Beach, CA, November 1973.
60. Ramamoorthy, C. V., and Kim, K. H., Software Monitors Aiding Systematic Testing and Their Optional Placement, Proceedings of the 1st National Conference on Software Engineering, IEEE Catalog No. 75CH0992-8C, September 1975.

61. Drummond, M. D., Jr., A Perspective on System Performance Evaluation, IBM Systems Journal, Vol. 8, No. 4., pp. 252-263, 1969.
62. Dahl, O. J., Dijkstra, E. W., and Hoare, C. A. R., Structured Programming, Academic Press, London, 1972.
63. Ryder, B. G., The FORTRAN Verifier: User's Guide, Bell Telephone Laboratories, Murray Hill, New Jersey, 1974.
64. Miller, E. F., Methodology for Comprehensive Software Testing, AD-A 01311, available from NTIS, February 1975.
65. McGowan, C. L., and Kelley, J. R., Top-Down Structured Programming Techniques, Petrocelli/Charter, New York, 1975.
66. Mills, H. D., On the Development of Large Reliable Programs, Proceedings 1973 IEEE Symposium on Computer Software Reliability, IEEE, New York, 1973, pp. 155-159.
67. Schindler, Max, ed., "Today's Software Tools Point to Tomorrow's Tool Systems," Electronic Design, July 23, 1981.
68. Houghton, R. C., Oakley, K. A., eds., NBS Software Tools Database, (U.S.) National Bureau of Standards, Washington, D. C., PB81-124935, October 1980.
69. Stucki, L., Automatic Generation of Self Metric Software, WD 2144, McDonnell Douglas Astronautics Company, Huntington Beach, CA, 1972.
70. Brown, J. R., Evaluating the Effectiveness of Software Verification-Practical Experiences with an Automatic Tool, AFIPS Conference Proceedings 41, 1972, NCC, Montvale, NJ, 1972.
71. Reifer, D. J., and Ettenger, R. L., Test Tools: Are They a Cure-All? SAMSO-TR-75-13, AD/A-004 078/2GA, October 15, 1974.
72. Data Base Design Aid, General Information Manual, IBM GH20-1626-1, IBM Data Processing Division, White Plains, NY, 1975.
73. FORTUNE User's Guide, IBM 360/370, CAPEX Corporation, Phoenix, AZ, 1974.
74. Aerospace Software Development Tools, Martin Marietta Aerospace Corporation, Denver, CO, 1972.
75. Deutsch, Michael, Software Project Verification and Validation, IEEE Computer, April 1981.
76. Jensen, Randall W., and Tonies, Charles C., Software Engineering, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1979.
77. Bersoff, Edward H., Henderson, Vilas D., and Siegel, Stanley, G., Software Configuration Management, an Investment in Product Integrity, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1980.

78. Controlling the Functional Testing of an Operating System, IEEE Transactions on Systems Science and Cybernetics, SSC-5 (1969).
79. Howden, William E., Functional Program Testing, IEEE Transactions on Software Engineering, Vol. SE-6, No. 2.
80. Howden, William E., Methodology for the Generation of Program Test Data, IEEE Transactions on Comput., Vol. 3-24, 1975.
81. Dijkstra, E., Structured Programming, Software Engineering, Brussels, Belgium, NATO Science Committee, 1967.
82. Nelson, TRW Symposium on Reliable, Cost-Effective, Secure Software.
83. Head, Robert V., Testing Real-Time Systems, Part 2: Levels of Testing, Datamation, August 1964.
84. Hetzel, Program Test Methods
85. Kelliher, D. W., Software Quality Assurance and Production Control Practices in the Acquisition of Large Systems, Technical Report MTR-6906, (McLean, VA: MITRE, 1975).
86. Fujii, Marilyn S., "Independent Verification of Highly Reliable Programs, Logicon, Incorporated, San Pedro, CA,
87. Haley, Anthony, "Test Plan/Test Report for MIL-STD-1553," SEAFAC, ENASD 81-1, May 1981.
88. Mark 33 Digital Information Transfer System (DITS), ARINC Specification 429-6, February 18, 1982.
89. Aircraft Internal Time Division Command/Response Multiplex Data Bus, MIL-STD-1553B, September 21, 1978.
90. Chun, Randall K., "ARINC 429 Digital Data Communications on the Boeing 757 and 767 Commercial Airlines," AIAA 81-2267.
91. Voight, James L., and Ellis, Donald H., MIL-STD-1553 Monitor Interface for Flight Test, 2nd AFSC Multiplex Data Bus Conference, ASD-TR-78-34, October 1978.
92. Boice, Michael, Digital Data Acquisition System (D-DAS) - A Flexible Tool for Simulation, Debug, and Traffic Monitoring of 1553 Systems, 2nd AFSC Multiplex Data Bus Conference, ASD-TR-78-34, October 1978.
93. Sottile, Michael R., Data Bus Simulation and Testing on the Space Shuttle Program, 2nd AFSC Multiplex Data Bus Conference, ASD-TR-78-34, October 1978.
94. Crossgrove, W. A., et al, Multiplex Applications Handbook, Prepared for Aeronautical System Division, May 1, 1979.
95. Notice 1 Military Standard Aircraft Internal Time Division Command/Response Multiplex Data Bus, February 12, 1980.

96. Szalai, Kenneth J., Jarvis, Calvin R., Krier, Gary E., Megna, Vincent A., Brock, Larry D., and O'Donnell, Robert N., Digital Fly-By-Wire Flight Control Validation Experience, R-1164, The Charles Stark Draper Laboratory, Inc., June 1978.

97. Archibald, Don M., The Role of Simulation in the Aircraft Certification Process, Federal Aviation Administration Report FAA-RD-77-17, January 1977.

98. O'Lone, Richard G., Two-Pilot 767 Transport Scheduled for First Flight, Aviation Week and Space Technology, March 15, 1982, p. 37-39.

99. Newcomb, Gary L., Higher Order Language (HOL) Impact on Hot Bench Testing, IEEE 1979 National Aerospace and Electronic Conference (NAECON), May 1979.

SECTION NINE
CURRENT VALIDATION PROCEDURES

TABLE OF CONTENTS

	Page
SECTION 9	9-1
9. CURRENT VALIDATION PROCEDURES	9-1
9.1 Design Validation	9-1
9.1.1 Theoretical Reliability Analysis	9-1
9.1.2 Electronic Design Verification	9-2
9.1.3 Failure Modes and Effects Analysis	9-2
9.1.4 Sneak Analysis	9-4
9.2 Hardware Testing	
9.2.1 Development Tests	9-6
9.3 Acceptance and Flight Assurance Tests Performance	9-7
9.3.1 Initial Acceptance Test	9-7
9.3.2 Flight Assurance Tests	9-7
9.3.3 System Acceptance Test	9-8
9.4 Software and Systems-Level Test Facilities	9-9
9.4.1 Digital Computer Emulation	9-9
9.4.2 Software Development Facility	9-10
9.4.3 Avionics Integration Support Facility	9-10
9.4.4 Iron Bird	9-10
9.4.5 Flight System in Aircraft	9-12
9.5 Software Verification	9-13
9.5.1 Quality Assurance in Design	9-13
9.5.2 Programming	9-14
9.5.3 Module Verification	9-15
9.5.4 Module Integration on Flight Computer	9-15
9.6 System Validation	9-16
9.6.1 Independent Verification and Validation	9-16
9.7 Piloted-Mission-Profile Testing	9-25
9.7.1 Aircraft Integration Testing	9-26
9.8 System Flight Test	9-27
9.8.1 High-Speed Taxi Tests	9-27
9.8.2 Flight Tests	9-28
9.9 Software Anomaly Experience	9-28

TABLE OF CONTENTS (Continued)

	Page
9.9.1 Software Modification Experience	9-29
9.10 Synopsis of Current Digital Validation Experience	9-30
9.11 References	9-31

LIST OF TABLES

Table		Page
9-1	FDIR Test Procedure (Reference 1)	9-17
9-2	FDIR Test Matrix (Reference 1)	9-19
9-3	Examples of Simulated Software Faults (Reference 1)	9-20
9-4	Summary of FDIR Test Results (Reference 1)	9-20
9-5	Stress Test Procedures (Reference 1)	9-22
9-6	Stress Test Results (Reference 1)	9-23
9-7	Piloted FMET Series (Reference 1)	9-24
9-8	Results of Piloted FMET Series (Reference 1)	9-25
9-9	Aircraft Integration Tests (Reference 1)	9-27
9-10	In-flight Computer and IFU Failure Experience (Reference 1)	9-28
9-11	Software Anomaly Experience (Reference 1)	9-29

SECTION 9

9. CURRENT VALIDATION PROCEDURES

Validation begins in the system design phase and continues throughout the operational life of the system. This section discusses validation procedures currently or recently used for present day digital flight control and avionics systems. These procedures were drawn from those used for the NASA F-8 Digital Fly by Wire (DFBW) (reference 1), the F-18 (reference 2), the YC-14 (reference 3), and the work to date on the Boeing 767 (references 4 and 5), and DC-9 Super 80 (reference 6).

9.1 DESIGN VALIDATION.

"In this stage, analysis is performed to determine if a design itself meets all the requirements, assuming the hardware is built as designed and has a normally expected component failure rates. Four of the steps done to validate the design will be discussed here. These include the theoretical reliability analysis, electronic design verification, failure modes and effects analysis, and sneak-circuit analysis." (Reference 1)

9.1.1 Theoretical Reliability Analysis.

"One of the first steps in validating the design of a flight critical system is to determine the approximate theoretical reliability of the system. This analysis is necessary to assure that the basic configurations chosen for the system have the inherent capability to satisfy the reliability requirements for the functions to be performed. This reliability analysis is performed by obtaining the approximate reliability requirements for the functions to be performed. This reliability analysis is performed by obtaining the approximate reliability of each of the subsystem's major component parts. These estimates are based on analysis and experience of the actual unit if it exists. If it is a new component, approximations are made based on the preliminary design of the unit and on the experience with similar equipment. The fundamental statistical and reliability equations are part of the particular system configuration being studied. These equations give the probability of system failure in performing the required operations as a function of the failure rates of the component units. The equations take into consideration the number of redundant channels, the number of channels needed by the redundancy management scheme, the reliability of the selftest, and other factors affecting the probability of total failure. The computed probability of failure is compared with the requirements for the system to determine the adequacy of the system configuration and associated hardware.

"Theoretical reliability analysis performs one of its most valuable roles in the configuration tradeoff studies. It becomes a major factor in determining the number of channels required, the methods used for redundancy management and failure detection, and in many other systems design considerations. Once a configuration is chosen, this analysis gives the reliability requirements for the component parts of the system. It is particularly valuable in determining the sensitivity of the total probability of failure to the reliability of critical parts. This process thus identifies where particular care must be exercised in the design, verification, and validation of the system.

"Theoretical reliability analysis can also be useful after the design is complete by being performed on the actual hardware that is built. This analysis would use a detailed system configuration implemented to give an updated estimate of the reliability actually expected for the test; it gives the baseline numbers in which the test results and actual operation of the system can be compared." (Reference 1)

Pitfalls in the theoretical reliability analysis include making simplifying assumptions which results in a higher theoretical reliability than is correct for the actual system (reference 1), and an over-reliance upon the use of MIL-HDBK-217C for calculation of failure rates for complex devices which can result in over optimistic estimates of reliability (references 1 and 7). All of these theoretical reliability analyses assume the components have the expected failure rates and there is no error in implementing the design.

9.1.2 Electronic Design Verification.

"Once it has been assured that the basic system has the theoretical capability of meeting the reliability requirements, it must be assured that the detailed design meets the system requirements with no design faults. In a digital system, both the hardware design and the software design are critical." (Reference 1)

In the F-8 DFBW, each circuit design was given to a design engineer with competence equal to the one doing the original design. "This engineer reviewed the design in relation to the requirements for that design. A design review was then held among the reviewing designers, the original designer, and the design supervisor to resolve any discrepancies." (Reference 1)

"Another aspect of design verification is to assure that the proper parts are used. The parts were reviewed by the reliability staff to assure that they were adequate for the application. Where possible, parts are used with a proven reliability history. However, in an advanced development program there is a desire to use components that are as close to the state-of-the-art as possible. In these cases, the proven history may not be as great as would be desired. Experience on devices is monitored through industry and government-reliability exchange programs to help assess the risk. Any device that has shown a problem is watched carefully.

"Of course the design is continually verified through each succeeding stage of the development program. The next test after the verification of the paper design is to build and test the breadboard prototype circuits." (Reference 1)

9.1.3 Failure Modes and Effects Analysis.

A Failure Modes and Effects Analysis (FMEA) is performed to assure that the design objective is met. In the case of the F-8, "This analysis was not performed at the detailed component level, but was a high-level analysis covering all units and signals in the system. Single-failure analysis was performed for:

- (1) Input failures.
- (2) Output failures including signals to actuators and pilot displays.
- (3) Interchannel communications failures.
- (4) Engage logic failures.
- (5) Power failures.

"There was also analysis of related dual failures. Failures in more than one channel were analyzed since multiple failures in one channel could, at worst, cause a loss of one channel. Multiple failures which were not related such that their combined affects could be obtained from the individual single failure tables also were considered. The dual failures covered were:

- (1) Dual serial I/O failures.
- (2) Dual input failures.
- (3) Dual output failures.
- (4) Dual interchannel communications failures.
- (5) Dual power failures.

"To the level of two failures, the digital system design objectives are realized. The software cannot be examined for generic failures with 100 percent confidence. Within this limitation, the FMEA did not reveal a flight-critical single-point failure." (Reference 1)

"The increased levels of reliability required are moving systems further and further away from levels that can be analyzed and demonstrated with conventional methods. Individual electronic units with failure rates in the range of 10^{-3} to 10^{-4} per hour can be analyzed by conventional methods such as FMEA's using established experience for component failure rates. The predictions made by these methods can be rather accurate and can be confirmed by actual service experience. A typical production unit will accumulate hundreds of thousands of operating hours per year and will experience many failures, giving a statistically significant estimate of its actual reliability. However, to achieve the very high reliability required of a flight-critical system, several units have to be integrated into a system which can tolerate faults and still operate. The reliability of this total system has to be assessed theoretically, based on an analysis of the reaction of the system to failures within the individual units. The reliability of individual units is an important input into this system analysis. Thus, conventional unit-reliability analysis will continue to be important. However, the methods used to combine this data to give the total system-failure rate are still being developed. We believe the contribution to the system failure rate due to combinations of random failures of individual components within units can be determined with theoretical (albeit complex) analysis techniques. We do not know of any method for demonstrating the absence of potential common-mode failures or generic design faults to this low level or probability. If a method were proposed, it is obvious that it could not be demonstrated by service and experience, since the goal of the design is to produce a low probability of one failure in whole fleets of aircraft over their entire life spans." (Reference 1)

"Another characteristic of emerging systems is a greatly increased complexity. This complexity is due both to the increased capabilities of the systems and to the necessity for redundancy and the associated redundancy management necessary to meet the reliability requirements. Conventional reliability-assessment methods can usually still be applied to these systems. However, the great increase in the number of different combinations of paths and conditions makes a usual detailed fault analysis very complex and difficult to perform. They are also difficult to understand; thus, their usefulness and cost effectiveness is reduced. New validation methods can make a valuable contribution to increasing the confidence that the system actually is able to meet its reliability requirements." (Reference 1)

Several reliability analysis techniques previously discussed in Section 8 of this handbook have been developed to aid in the analysis of complex systems. "General analysis techniques such as CAST and CARSRA often prove to be difficult to apply to actual specific systems. More often than not, there is some system characteristic that is not well modeled by the general method. In most cases, systems are most efficiently analyzed by programs specifically tailored to that particular system. These general techniques, however, do provide insight into the failure characteristics of advanced systems and provide a baseline from which the detailed analysis of specific systems can be developed." (Reference 1)

"Theoretical analysis and simulation can only provide an aid in support of the validation process. Theoretical analysis can derive the system failure rates due to the expected failure rates of components. Redundant fault-tolerant systems are purposely configured to account for these faults, and can be designed to achieve an arbitrarily low failure rate due to this source of failure. Although difficult, the problem of determining this system failure rate is a solvable problem, and these techniques help solve the problem. Of more concern are specification faults, design errors, and induced failures. Analysis and simulation methods have very little ability to detect these types of faults." (Reference 1)

9.1.4 Sneak Analysis.

Boeing Aerospace Company has developed a computer-aided analysis technique for electrical and electronic hardware systems, known as sneak circuit analysis. They have extended this technique to software through the use of electrical-software analogies (known as sneak software analysis). They have further extended the technique to "integrated" analysis of a system software and hardware using the term Sneak Analysis (SA).

The sneak circuit analysis was developed to detect latent circuit conditions that may cause unwanted functions or inhibit desired functions. "These sneak circuits can cause system failure that is not the result of the component failure. There may be four basic types of sneak circuit:

- (1) Sneak Paths, which cause current or energy to flow along an unexpected route.
- (2) Sneak Timing, which may cause or prevent the flow of current or energy to activate or inhibit a function at an unexpected time.
- (3) Sneak Indications, which may cause an ambiguous or false display of system operating conditions.
- (4) Sneak Labels, which may cause incorrect stimuli to be initiated through operator error.

"The data used to perform this analysis is electrical continuity information based on wire run list and electrical schematics. Support data in the form of assembly drawings, overall system level interconnect diagrams, and electrical schematics describing the interface of out-of-scope components was also used for the analysis." (Reference 1)

"The computer program searches for a continuity path and generates reports from which network trees can be drawn. These network trees were drawn at the topological form which facilitates the application of sneak circuit clues.

Hardware design concerns considered in the sneak circuit analysis include:

- (1) Single failure points.
- (2) Unnecessary components/circuits.
- (3) Redundancy improperly implemented.
- (4) Unnecessary power consumption.
- (5) Improper component biasing.
- (6) Increased noise susceptibility.
- (7) Improper applications of components.
- (8) Improperly terminated integrated circuits.
- (9) Voltage transients.

In addition, the sneak circuit condition lists discrepancies found in any schematics or documents.

The sneak software analysis (SSA) tests for:

- (1) Missing logic.
- (2) Unused logic.
- (3) Invalid wait loops.
- (4) By-passed logic.
- (5) Open-ended logic.
- (6) Logic loops.
- (7) Conflicting tests.
- (8) Invalid outputs.

The software design concerns considered include:

- (1) Improper sequence of instruction.
- (2) Unnecessary logic.
- (3) Unreferenced labels or variables.
- (4) Redundant logic.
- (5) Difficulties with future maintainability due to inconsistencies between software specifications and code.

Document errors such as discrepancies in any two specification documents are included in the reports generated by the Sneak Software Analysis.

The minimum data required for the Sneak Software Analysis includes:

- (1) Compiler and/or assembly listing.
- (2) Reference manuals for computer, language, and operating systems.
- (3) Requirements and/or specifications of software design and hardware/software interface control.

Other desirable data includes program documentation such as systems descriptions/requirements, module descriptions, flow diagrams, and data structure definitions/description." (Reference 8)

Reference 8 states "a computer-aided approach is necessary for large-scale programs, to allow organization of the data base for analysis and to avoid human errors. The computer is capable of sorting the data, identifying all current and logic paths, and of providing accountability for the sequency of elements in these paths."

General Dynamics also performs sneak analysis using pathfinding. Manual pathfinding becomes inadequate as system complexity grows and "falls apart" under the heavy load of digital circuits in microprocessors that characterize most new systems. General Dynamics computer aids are called the Pathfinding Programs (PFP) (Reference 9). "These programs are primarily aids to tracing paths through complex networks, and trained analysts still are needed to examine the networks to find the sneak conditions. For example, the pathfinding programs assume that all switches in a current path are closed, but the analyst must determine which paths actually can be active at any given time." (Reference 9)

Boeing states "only by (1) using the network trees and (2) applying the clue lists can trained engineers discover sneak conditions. The network trees and clue lists represent a systematic, logical, consistent approach to identifying sneak conditions." (Reference 8)

9.2 HARDWARE TESTING.

Hardware testing is conducted during the development phase, upon individual components to assess the performance of the design at the subsystem level and finally at the integrated system level.

9.2.1 Development Tests.

"These tests are, as a rule, devised by the contractor with prior approval at the 'plan' level, and consist of tests designed to assure the development and fabrication of the system." (Reference 1)

9.2.1.1 Component Tests. Component testing varies greatly depending upon several factors. These factors include:

- (1) Type of testing performed by the manufacturer.
- (2) The difficulty of replacing the component during later stages of testing.
- (3) The adequacy of testing taking place in later stages of development.
- (4) Importance of performance of the component.
- (5) The cost of the component.

If integrated circuits and semiconductor devices are purchased to conform to MIL-STD-883B level B, where possible, it is probable that the incoming acceptance testing will not be as extensive as that for commercial components. In most aerospace applications, the manufacturer either procures components qualified to military environmental levels or performs extensive incoming testing to screen the components (reference 1).

9.2.1.2 Design Evaluation and Performance. "Design evaluation tests began as rudimentary tests early in the design stages and continued throughout the program, becoming most important during the build and operation of the first system breadboard. The objective of the design and evaluation tests was to determine the

adequacy of the design to meet system requirements. The most significant portion of the tests relating to system redundancy was the failure modes and effects test and its relation to the failure modes and effects analysis." (Reference 1)

9.2.1.3 Subsystem Test Performance. These type of tests generally test the individual subassemblies and their interfaces with each other. Tests are designed to verify the correct internal operation of each subsystem and to provide assurance that the systems (breadboard, iron-bird, and flight) were functionally identical. "Wiring errors have been found in the iron-bird system that were not uncovered during subsystem or integrated system testing, which lends support to the idea, no matter how well conceived or carried out a test is, it is still fallible. Testing, in general, and particularly at the subsystem and integrated levels, needs to be well thought out and thoroughly reviewed by design and system personnel to minimize oversights." (Reference 1)

9.2.1.4 Integrated System Test Performance. "The system-integration testing was performed in two stages:

- (1) Operation of system with test software and static data.
- (2) Operation of system with simulated cockpit and hybrid-flight simulator.

Initial tests of the system used simulated aircraft interfaces and test software to verify correct interface operation. Preliminary tests of system response to transient power failures and loss of redundancy due to hardware failures were performed." (Reference 1)

"Following initial testing, to show correct system hardware operation, the system was interfaced with a simulated cockpit and a flight-simulation facility to verify operation of the system in a simulated flight environment. The responses of the system to external signal interruptions and transient power failures was determined." (Reference 1)

9.3 ACCEPTANCE AND FLIGHT ASSURANCE TESTS PERFORMANCE.

"The acceptance and assurance tests are contractual requirements. The details of the test plans and procedures are devised by the manufacturer, and are reviewed and approved by the user. These tests are intended to show that the system meets its requirements as detailed in the statement of work." (Reference 1)

9.3.1 Initial Acceptance Tests.

The initial acceptance tests are performed at the manufacturer's facility and use simulated interfaces. These tests are designed to demonstrate that the component or system meets the requirements of the specification and that all interfaces are correct.

9.3.2 Flight Assurance Tests.

Flight assurance tests, which are performed at the manufacturer's facility, are designed to test the operation throughout the expected aircraft physical environmental range of vibration, temperature, and altitude. These tests are often designed to show that the equipment meets the appropriate levels called out by the Radio Technical Commission for Aeronautics (RTCA) in DO-160A (reference 10). As stated in that document, "the environmental conditions and test procedures defined

herein are only intended to determine the performance of the airborne equipment under these environmental conditions and are not intended to be used as a measure of service life of the airborne equipment under test."

9.3.2.1 Temperature and Altitude Tests. As stated in reference 10, "the selection of a temperature/altitude category depends on the location in (or on) the aircraft, the maximum operating altitude of the aircraft, and whether the equipment is located within a temperature and/or pressure controlled area. The above conditions must be taken into consideration by the equipment designer in evaluating these requirements as determined by the end use of the equipment." Specific test procedures may involve a combination of automatic testing with automatic/manual monitoring and manual testing with manual monitoring. In addition to the system operating in its normal modes, the test designer may require injection of failures to demonstrate correct operation of the failure-monitoring software during these environmental tests.

9.3.2.2 Vibration Tests. Sinusoidal and random vibration tests are described in reference 10. Depending upon the vibration test equipment, a Combined Environment Reliability Test (CERT) may be performed during which the device under test is subjected to a vibration environment while undergoing temperature and altitude, humidity, input voltages, and on/off cycling in an attempt to replicate the environment mission profile as described in reference 11. "Current reliability test procedures do not expose avionics equipment to realistic environmental stresses, thus contributing to the poor correlation between field and laboratory failure rates and modes." (Reference 11)

9.3.3 System Acceptance Test.

"The system acceptance test consisted of three parts:

- (1) Functional demonstration
- (2) Failure modes and effects tests
- (3) EMI.

The above tests were performed with the flight system installed in the ironbird at DFRFC. The failure modes and effects test was performed with the breadboard system installed in the iron-bird facility." (Reference 1)

9.3.3.1 Functional Demonstration. "The functional demonstration was intended to show correct operation of the system installed in the iron bird by:

- (1) Operation of subsystem assemblies
- (2) Operation of system using aircraft cables for subsystem interconnections
- (3) Interfaces with other aircraft interfaces." (Reference 1)

9.3.3.2 Failure Modes and Effects Test. The iron-bird simulation facility permits piloted simulation using the operational flight programs and special test software to verify the unfailed system performance, failure modes and effects test, stress test, and piloted failure modes and effects test. These tests are described in greater detail in later paragraphs of this section.

9.3.3.3 EMI Test. These tests include the:

- (1) Induced signal susceptibility tests (reference 10)
- (2) Radio frequencies susceptibility tests (radiated and conducted) (reference 10)
- (3) Emission of radio frequency energy test (reference 10).

9.4 SOFTWARE AND SYSTEMS-LEVEL TEST FACILITIES.

There are a variety of test facilities available for verification and validation of digital flight control and avionic systems. These facilities include a digital computer emulation facility, software development facility, "hot bench," iron bird simulation facility, and flight line facilities.

"In addition to major test facilities, some fundamental tools were also available for software development in digital computer testing. There were:

- (1) Assembler and link editor support software for producing flight code and load tapes
- (2) Computer test set for software and hardware debug
- (3) Function test program (FTP) for thorough testing of the digital computer
- (4) Real time software for on-line debug of software
- (5) Tape drive, CRT, and printer for load, verify, and test." (Reference 1)

In addition to these support facilities, the modern Avionics Integration Support Facility will include multiple processors for control of the testing and data acquisition and reduction. These processors often host the compiler for the flight computer's source code and generate the object code in a loadable format depending upon the particular flight computer.

9.4.1 Digital Computer Emulation.

A number of recent programs involving the development of digital flight control and avionic systems have used emulations of the flight computer (references 1, 3, and 12). The digital computer emulation is a software program that resides on a host processor and emulates in software the operation of the target computer. The digital devices of the processor are emulated in software at the gate-level and the software program computes the output of each device, in sequence, at the initiation of the appropriate clock pulse. At the completion of a single pass, the emulators' responses are in one-to-one agreement with those of the target hardware. The emulator can be used to run test flight software even before the actual flight computer is available. "The emulator provides an instruction-by-instruction simulation of the flight computer's processing of a flight software. This allows the programmer to trace the exact execution sequence and to examine all intermediate computational results." (Reference 1)

An emulator does not necessarily operate in real-time and often operates many times slower than real-time. In addition to using the emulator to develop and debug the software, an emulator can be used to test the operation with injected faults. One problem with this use is that the failure mode data, at both the function and gate-levels, is generally not available (reference 12). A detailed emulation of a digital device at a level at which failure mode data is not available is of limited use.

"The essential purpose of emulation is to provide a systematic and comprehensive treatment of fault analysis for digital systems. The technique can be used in a variety of applications such as in the design and validation of a self-test procedure. It is customary to specify that self-test or BIT procedures should have a specified fault detection coverage. Emulation can be used to design an efficient test and to validate the results. In this connection, it is noted that the coverage requirement does not normally specify whether the faults are at the gate-level or pin-level" (reference 12).

9.4.2 Software Development Facility.

Software development facilities usually contain the flight computers as well as various host processors for development of the operational flight program. The F-18 software development facility consisted of "two flight control computers, a test bench to access the flight software, and an interface/actuator model bench containing sensor and control surface models having the same dynamic response and avionics interface as the aircraft units. This facility also interfaces with the MCAIR flight simulator, F-18 mission computer, and head-up display, thereby providing complete closed loop operation and enabling pilot evaluation of flight response and output displays" (reference 2).

The NASA F-8 program software development facility was used at the contractor's facility for software and interface hardware development. It provided "early real-time experience on the flight software before all the flight computers were delivered. This test bed provided a controlled interface for initial systems tests. This kind of a system, however, generally cannot be used with confidence to identify and fix problems associated with timing or intersystem communication. Outside of strictly single-string computations, it is never possible to say for sure what tests have validity when run on this system" (reference 1).

9.4.3 Avionics Integration Support Facility.

These facilities generally contain flight computers, test bench, interface/actuator model bench, multiplex/mission computer, and in some cases, rate and accelerometer sensor rate tables (references 2, 3, 5, 6, 13, through 15). In some cases, the avionics integration and support facility is colocated with the iron bird such as Boeing's Digital Avionics and Flight Controls Laboratory at its Renton Flight Simulation Center (reference 5). The digital avionics and flight control laboratory at Boeing consists of nine work stations which can support tests ranging from stand-alone line replaceable units software validation to subsystem integration tests. Multiple test stations can be interconnected and supported by the simulation host computers for integration testing. Special test drivers were developed for the simulation host computers to automatically perform extensive test routines on the flight management computer software. Simulations of the airplane, engines, sensors, and other subsystems were prepared and used to perform realistic, real-time tests of the FMCS. Both open-loop and closed-loop tests with simulated autopilots and autothrottles are conducted using the laboratory facilities at Boeing (Reference 6).

9.4.4 Iron Bird.

The "iron bird" simulation facilities vary in capability but normally permit piloted closed-loop dynamic simulation of the aircraft using some combination of real sensors and real actuators as well as simulated sensors and actuators. These facilities normally use six degree-of-freedom equations of motion (reference 6).

"The total simulation complex is oriented to a single concept, namely that actual hardware to be flown in the vehicle is operated with a full aircraft and control system simulation in real-time" (reference 6).

"During the 3 year development phase of the DC-9 Super 80 program, software was developed, verified, and validated on this facility. When preliminary coding was complete, integrated software was exercised in prototype hardware on the validation facility to performed closed-loop real time execution of mode selection, performance, annunciation, and logic. In the process of developing and testing the software, many elements of the classic software testing were accomplished. Scaling accuracy, verification of overflow protection, determination of expected performance for normal inputs, and response to abnormal inputs or failed logic states were all tested routinely as part of the software functional checkout. Once the executive and fundamental data handling software was completely tested, the functional modules were added as developed and tested at an integrated level in real-time" (reference 6).

"In addition to its use for the development of the digital flight guidance system (DFGS), the Validation Facility played a significant role in obtaining material to support certification of the digital flight guidance system. Since the software being used in the computers in the facility was identical to that being flown in the developmental flight test, it remained only to show fidelity of the aircraft's simulation to performance of the actual aircraft in establishing its credentials for demonstrating DFGS performance. This task was accomplished by correlating time histories of aircraft open loop performance with that of the simulation. Once close correlation was accomplished, accreditation of the facility was established and testing for certification proceeded" (reference 6).

"One major test program established the effectiveness of the system monitoring incorporated in the software. The DFGS contains 10 separate monitoring techniques for failure detection of:

- . Sensors which interface with the system
- . Servos interfacing with the system
- . Internal functioning of the computer I/O complex
- . The computer memory
- . The processor and its control store.

"The validation facility operating in real-time with either actual sensors or realistic sensor models made an excellent vehicle to study effects of failure in each of these areas. We were also able to determine the residual effects upon performance after a failure was detected and appropriate action taken by the system. This process was started by conducting sensitivity studies of the effects of various failures in their different forms. The classic hardovers presented essentially trivial considerations. Ramp failures of variable magnitude, rate, and limit were much more subtle. A third variant, time insertion of a failure and the duration of its insertion, presented greater challenges. By modifying the software handling various input or output signals, we were able to readily control the timing, type, and magnitude of failure insertions. Presentation of failure effects in real time with specific quantified impacts on autoland performance enabled us to make specific changes wherever deficiencies were identified or to demonstrate effectively that the results were benign" (reference 6).

"Additional tests conducted for certification credit concerned the effects of failures of the aircraft control system and the ability of the DFCS to monitor for these failures. Among others, these failures included elevator tabs disabled or surface frozen, loss of spoilers or control on one wing, and malfunction of the ground sensing mechanisms in the aircraft. In the case of a disabled surface, it was necessary to determine that autopilot monitors detected the aircraft failure as soon as it occurred, disengage the autopilot, and warn the pilot. For failures that were not detectable, it was shown that their effects were benign. In every case, the results of failure insertion and its detection or lack of detection were graphically displayed in real time histories for all significant parameters. With these results, it was possible to show compliance with the safety requirements of FAR 25.1309 or the performance criteria of AC 20-57A. The effects of these failures in the presence of various detection techniques could only be observed and analyzed effectively in this simulation environment. A significant benefit of these studies was that the entire set of failures and their effects could be studied, and a subset of most critical cases chosen for demonstration on the aircraft. Once the fidelity of the simulation is established, this is a valid approach" (reference 6).

In the NASA F-8 DFBW program, "The iron-bird facility was the key element in the system verification, validation, and flight qualification. It exposed the real-system problems, allowed subsystem interface problems to be resolved, and especially provided a high degree of confidence in the verification and validation tests results. The iron bird also provided the all-independent pilot interface" (reference 1).

The iron-bird facility was used to test the actuator interfaces with the flight control computers in the F-18. "Closed loop software response with the airframe dynamics was evaluated in the software development facility and with the iron bird. These tests evaluate a response and damping effectiveness of the control augmentation system modes" (reference 2).

"It should also be recognized that an iron-bird facility is expensive to construct and to maintain, being more like an airplane than a simulator" (reference 1).

9.4.5 Flight System in Aircraft.

"The aircraft itself is an important test facility, for only in the vehicle are all the subsystems in their true flight configuration. Final flight-readiness tests are best accomplished on the vehicle itself. If iron-bird testing has been properly accomplished, only higher level system tests need be performed on the vehicle. The primary disadvantages of using the aircraft for a substantially more than final system readiness tests are that the all-up airplane is available quite late in the development cycle, it is more difficult to tie in special test instrumentation, and there is a limited amount of time available on the aircraft for systems testing."

"The flight-test program is also part of the system validation process. The flight environment provides those unmodeled characteristics that are not included in the simulation. The hardware itself is exposed to simultaneous temperature, vibration, and operational situations which never seem to be covered in ground-test matrices. The pilot/crew interface with the system is stressed by mission or task factors that are not reproduced in ground simulation."

"In the case of a flight-critical digital flight control system, the flight regime represents a very hazardous verification and validation environment. Obviously, the core system must already be qualified to a high degree of confidence or must have a safe fall-back capability, or both, in order to make the first flight."

"The flight environment, while providing the only truly credible qualification results, has obvious drawbacks as well. There are generally only a small number of test hours, visibility into the system operation is limited by instrumentation capability, and the cost is very high" (reference 1).

"The preflight test program was verified on the iron bird, but was validated and qualified on the aircraft because of the need to show the proper operation with the airplane configuration. The servo electronics/actuator interface was performed on the airplane principally to off-load work on the iron bird, which was heavily committed to software and digital system testing. The electrical and hydraulic systems were laid out and tested initially in the iron-bird, but of course, had to be qualified in the aircraft itself" (reference 1).

"Although all sensor I/O and redundancy management software was verified/validated/qualified on the iron bird, the hardware interfaces were also tested and qualified on the airplane itself. Because of costs, the iron bird was not normally operated with actual sensors, and had no provision for an actual air-data interface. Special closed-loop resonances tests were also performed on the airplane in which structural mode excitation margins were determined. This required an all-up system and sensor configuration, which was available only on the actual vehicle" (reference 1).

"The only delivered validation test slated for the flight-test phase of the program were the sensor failure-detection threshold tests, which are dependent on actual sensor-output characteristics in the flight environment" (reference 1).

9.5 SOFTWARE VERIFICATION.

"Software verification — the determination that the generated code correctly performs its intended function — ideally begins the design phase. The software program consists of many modules of varying size, which together accomplish the total intended purpose of the program. If the individual modules are adequately defined and controlled from the design phase on, the verification is made less complex and costly in both time and expense" (reference 1).

9.5.1 Quality Assurance in Design.

"Quality assurance in design depends upon two important factors:

- (1) The software specification document, and
- (2) Programming ground rules.

Careful and full attention of these two factors will greatly ease the complexity and time necessary to perform software verification" (reference 1).

9.5.1.1 Software Specification. "The interpretation of the requirements must be clear and straight-forward and mutually agreed to by all parties involved. All changes or additions must be tightly controlled and documented. All approved changes must be published and communicated to all programmers working on code generation " (reference 1).

"The requirements specification and analysis phase is crucial in software development. If the system being developed is poorly thought out and poorly structured, no amount of effort and capability in later phases can make the project a success. On the other hand, an accurate, clear, concise and noncontradictory statement of the software requirements lays a firm foundation for later phases. If the requirements are specified in a manner to which the implementation can be directly tested, even further benefits can accrue. On any non-trivial project this is almost impossible to do manually, however. Several requirements statement languages which are computer processible have been developed to automate this process. The computer forces order and provides a much better cross-referencing and mapping facility than could ever be achieved manually" (reference 17).

"The design phase is required to determine the software structure and mechanization which will implement the requirements. While this is basically a creative process, the computer can again be used to advantage by storing and presenting data providing cross-reference services. Simulations using formal simulation languages are also very valuable during the design phase. Several program design languages exist which allow the top-down development of program structure and content in an easily manipulated form. Application of such languages can help ensure that the design meets the requirements and meets them in an efficient and easily maintainable fashion" (reference 17).

9.5.2 Programming.

"The implementation phase of software development is the phase when code is actually written and debugged. The following types of tools can be provided: Editors, debugging aids, source formatters, file control systems, simulators, compiler writers, compilers, meta-assemblers, assemblers, linkers, and loaders. Although basic forms of these tools are commonly used, improved tools in this area can lead to even greater productivity gains and improved adherence to the design" (reference 17).

"The establishment of programming ground rules is essential to the production, testing, and modification or updating of generated code.

"The first and foremost rule is: No code must be written until a logic diagram has been generated. As in the building of hardware, one needs blue prints to correctly build a piece of equipment. A programmer needs a logic-flow diagram to correctly code and document a particular computer function.

"The specification of maximum module entry and exit points (no more than two of each), self-contained temporary storage, and minimum external references insure that each module will be self-contained to the maximum level, and, once tested and proved to perform its function, will not be altered by the addition of another module to the developing program. Each module must be single purpose in function and that function must be carefully and fully specified. The execution of the module's function must be independent of other modules, except for common library routines which may be called by the module, and all necessary manipulation of arguments must be self-contained within the modules. The number and form of entry and exit arguments must be specifically called out and identified.

"The development and use of 'macros' to accomplish common subroutines in each module again reduces the level of testing, which must be performed to verify each module.

"Most computers, unless specifically designed to meet a given functional role, have an instruction repertoire far in excess of coding requirements. Many of these instructions are complex in execution and require extreme care in use. Totally eliminating or tightly controlling the use of complex instructions will make program verification and debugging less complex and costly. Straight-forward programming may cost some penalty in core requirements and execution time, but save considerably in the long run" (reference 1).

9.5.3 Module Verification.

"The test and validation phase of software development can also benefit from tools. This phase should demonstrate that the software system has no errors and satisfies requirements. Formal program analysis tools exist and are available to aid in test case development, fault isolation, and verification that all statements and branches have been executed. Other tools are available to store test cases and test history in a data base to document test completion and support regression testing. Embedded computers are essential to the operation of modern avionic systems. Potential loss of missions or aircraft can occur if errors exist in the software. Improved test tools can lead to increased confidence in the software, lower failure rates in the field, and decreased need for expensive software flight tests" (reference 17).

"The verification of a module's function begins prior to code generation. The first step is a construction of the logic-flow diagram, which basically describes the module's functions and serves as a programmer's guide in writing the module code. The generated code is checked against this flow diagram by both the programmer and some other programmer, which both verifies the logic diagram and code generated" (reference 1).

Modules may be tested on either a computer emulator or by execution in the actual target computer. "When testing execution of module code, extreme care is required to insure that any dummy routines used in the testing of the module do not cover up some error in the module code" (reference 1).

9.5.4 Module Integration on Flight Computer.

"Tools and procedures are available to support the integration of software with the deliverable hardware and the test of the integrated system which occurs in the system integration phase. Appropriate procedures, communication packages, and interface handlers are essential to the proper accomplishment of this function. Support should include communication from software development systems to/from the target computer system being integrated or tested. Ultimately, the tool host should be sufficiently available and accessible so that integration testing and documentation can be conducted as if all the resource of the tool host are available during integration of the system" (reference 17).

"When the program library contains a sufficient number of verified modules to begin generation of an integrated program, the program library then initiates production of the program. The program generated will contain the verified modules and any dummy routines necessary for program execution. The program is then executed in the flight computer, preferably in a full-system configuration. Extensive use is made of available ground-test equipment to ensure that the program execution is correct and as planned.

"The execution of a program on a computer emulator is not recommended as this does not fully and truly simulate actual target computer or system operation. An emulator is not normally capable of generating all the random inputs and time slicing which occurs in the actual system configuration. This factor is extremely important which dealing with multi-computer systems. On the F-8 program, the inability to test software changes on a multi-computer system at CSDL prior to release has proven troublesome. A program may execute correctly in a single computer and fail miserably when exposed to the randomness of operation of the actual multi-computer configuration" (reference 1).

9.6 SYSTEM VALIDATION.

"The system validation tests are performed on hardware and software that are essentially operational; i.e., all major hardware and software components are functioning in a nominal manner. This is important because every minute of operating time is an implicit test, exercising hardware and software interfaces hundreds of times. During this period of close scrutiny, it is critical that as much of the system as technically possible be operating, so as to expose deficiencies that occur during the test phase, but outside of the test plan. System validation is accomplished by carrying out several different types of tests" (reference 1).

9.6.1 Independent Verification and Validation.

On the NASA F-8 program an independent test team comprehensively verified individual functions of validated major system operation. Many of the tests conducted were duplicates of those accomplished by the programming team. "A key difference in these tests was that they were carried out in the environment of a fully integrated and nominally operational system on the NASA iron-bird facility" (reference 1).

"By its very nature, the verification and validation testing of the multi-computer system emphasizes abnormal operation because normal operation reveals nothing of the true characteristic of the system. A fact that a triplex system operates in the synchronized condition for many hours does not in any way indicate that it can survive even the most elementary fault or that it can provide any more fault tolerance than a single-channel system. In fact, with regard to multi-channel performance, normal quiescent operation yields data only on the nuisance faults statistics of this system" (reference 1).

Multi-channel system validation test approaches fell into four major categories for the NASA F-8 program. "The first category of testing established nominal operation. The second category of tests involves a systematic verification of the Failure Detection, Identification and Recognition (FDIR) software and also the Failure Modes and Effects Test (FMET). There is an overlap in these tests. The stress tests are conducted in order to evaluate system operation in configurations and conditions which fall outside of the controlled verification tests. In many cases, this involves reducing system operating margins to zero to establish ultimate breakdown boundaries.

"The piloted fault injection tests seek to verify that normal pilot response to fault conditions do not adversely couple with automatic FDIR actions. It is also essential that the system response to real faults, which occur during the test program, be analyzed. Each such fault is an extremely important test because real faults occur in circumstances and sequences not always considered in formal

testing. In the F-8 program, each such case was intensely analyzed to evaluate the system FDIR. As it turned out, there was ample opportunity to observe the FDIR process for natural faults, due to early problems with computer reliability.

"The actual experience gained during these tests, and the implications for future applications are described in the following" (reference 1).

9.6.1.1 Unfailed System Performance Tests. These tests were designed to establish a nominal multi-channel performance of the system and test its synchronization, timing, and intercomputer communication integrity.

Special test software is often used to measure the execution time of major modules of the program. A special software routine computes execution time by measuring elapsed time and then subtracting overhead time. "Timing figures were compiled over a long period of time, with both the average and maximum time noted" (reference 1).

"During this early test phase it was frequently observed that a channel was systematically losing sync, or that a channel, or occasionally all three channels, would fail to achieve sync at turn on. The implicit testing was to generate the majority of software design changes" (reference 1).

9.6.1.2 Failure Modes and Effects Tests. In the NASA F-8 program, "FMETs are designed to systematically verify that the digital system is fail safe and operational after a single disabling channel fault and fail passive after a second channel failure. These tests were performed after isolated software modules have been verified for correct functional operation. Table 9-1 summarizes the FMET strategy for both hardware and software faults.

TABLE 9-1. FDIR TEST PROCEDURE (Reference 1)

Inserted Fault	Verification Steps
Transient fault in one or more channels	<ul style="list-style-type: none">. Successful restart. Continued operation. No permanent faults. No adverse surface motion
Permanent fault in one channel	<ul style="list-style-type: none">. Restart attempted. Channel declared hard failed. Proper internal reconfiguration. Annunciation to pilot. System operational. No adverse surface motion
Permanent, like faults in two channels (successive)	<ul style="list-style-type: none">. Restart attempted. Auto transfer to bypass system. Proper internal status. No adverse surface motion

"The selection of the fault matrix presents a very difficult problem for a system such as this. It is apparent that a component-by-component or wire-by-wire open/short fault test matrix is impossible. The parts count and number of failure modes precludes even a modest attempt at verifying FDIR operation in this manner. Thus, the strategy used was to inject faults at the functional level and judge integrity on the basis of FDIR response to these functional faults. An important thing to note at this point is that the system must be designed at the outset to be testable. If the FDIR scheme is extremely sophisticated and involves operations on a large amount of data and a large number of intricate steps to detect and isolate faults, it is quite possible that the system will be unverifiable. That is, the number of tests required to establish correct response, even for those faults that can be thought of, exceeds the test capability available.

"The F-8 DFBW FDIR design and FMET approach are based on the following assumptions:

- (1) The ability of a channel to maintain synchronous operation and to communicate with its partners is a primary indicator of channel health

- (2) A relatively small amount of information exchanged between channels can be used as a further basis for this determining channel health. This information can be thought of as vital signs

- (3) Fault detection, isolation, and reconfiguration can be based on the manifestation of a fault, rather than on the detection of the fault itself

- (4) The restart mechanism will provide automatic transient fault protection without having to identify the source of the fault.

These assumptions dramatically reduce the test matrix. It is now possible to approach the failure modes and effects tests in the following manner:

- (1) Verify that the FDIR response to abnormalities in the small number of vital signs is correct.

- (2) Verify that all built-in test hardware produces warning signals to the FDIR for the types of faults they are designed to detect.

- (3) Verify that the restart process restores normal operation for transient abnormalities in the vital signs or for the built-in test warning signals.

- (4) Verify that the restart mechanism will suspend its attempt to restore operation for unrecoverable faults.

- (5) Verify the fail-operational fail-safe performance to all permanent faults.

These assumptions and observations produced a relatively modest test matrix which provides an extremely thorough coverage of the functional validation of the FDIR process. The test matrix is tabulated in summary form in table 9-2. No assumptions were made relative to the similarity of channels. That is, faults were induced for all combinations of two channels.

TABLE 9-2. FDIR TEST MATRIX (Reference 1)

Location of Injected Faults	Approximate Number of Faults	Types of Faults Injected
Interface Unit	200	<ul style="list-style-type: none"> . I/O data lines . Crosslink communication lines . Synchronization lines . Loss of entire card . Loss of entire connector . Loss of power
Encoder/Decoder	400	<ul style="list-style-type: none"> . Data lines . Strobe lines . Clock lines
Computer Hardware	20	<ul style="list-style-type: none"> . I/O connector . Spurious interrupts . Loss of power
Computer Software	150	<ul style="list-style-type: none"> . Faults producing program interrupts . Faults producing machine interrupts . I/O communication errors . Vital sign errors

"Breakout boxes were used to insert the majority of encoder-decoder faults and some of the interface unit faults. For most of the interface unit fault injection tests, however, special hardware was used which was built-in specifically for the failure modes testing. This hardware provided the capability to break signal paths or to simulate fail-high or fail-low conditions.

"The computer test set was generally used to insert software faults, although some special-purpose software was required to simulate certain faults. Table 9-3 lists in more detail the types of software faults introduced.

"The results of these FMETs are summarized in table 9-4. A remarkably small number of anomalies occurred, with only one being serious: The input data stream double fault. The encoder/decoder faults affects this display operation in a benign manner. The key observations are that the fail-operational fail-safe requirement was demonstrated in all but one condition. Although stress testing was to expose several restart recovery problems, it is significant that the system operation was verified to the extent demonstrated. The second item of significance was that all transient faults introduced were survived by the system" (reference 1).

TABLE 9-3. EXAMPLES OF SIMULATED SOFTWARE FAULTS (Reference 1)

Faults Introduced	Result
<ul style="list-style-type: none"> . Illegal-operation code . Privileged instruction . Fixed-point overflow . Significance test . Unnormalized floating-point operation . Store protect violation . Exponent underflow and overflow . Divide check 	Program Interrupt
<ul style="list-style-type: none"> . CPU and memory parity error . Timeout (No-Go discrete) 	Machine Interrupt
<ul style="list-style-type: none"> . Discrepancy in channel-identification tag . Discrepancy in computer time . Discrepancy in computer functional mode . Recognition of restart request . Critical error in crosslink transmission list . Error in self-test program 	Direct Restart

TABLE 9-4. SUMMARY OF FDIR TEST RESULTS (Reference 1)

Anomalies
<ul style="list-style-type: none"> . <u>Synchronization</u> <ul style="list-style-type: none"> . Continued operation for some sync faults
<u>Input Data Stream</u>
<ul style="list-style-type: none"> . No CBS down mode for dual input data line loss
<u>Encoder/Decoder</u>
<ul style="list-style-type: none"> . One type of first fault not detectable . Software wrap test incorrect
<u>Computer Software</u>
<ul style="list-style-type: none"> . Some faults detected by means not predicted
Observations
<ul style="list-style-type: none"> . System was fail-operational for every first fault . System was fail-safe for all but one second fault . All injected transient faults were survived

In the case of the DC-9 Super 80 "the design of the DFGS for the DC-9 Super 80 required that it provide fail-passive operation during critical approach/landing conditions through use of redundant sensors and a high level of self-monitoring. Concurrently, the redundant sensors were to be used as a means of enhancing system availability for noncritical cruise modes. Redundancy requirements and their reversionary capabilities are specified in the Failure Mode and Effects Analyses (FMEAs) submitted as part of the certification documentation. The total simulation facility was brought into play to demonstrate to the FAA that reversion modes and appropriate annunciation of the reversion occurred as intended for the system implemented. These tests concerned the auto throttle/flight director, autopilot/speed command system, yaw damper, Mach trim compensator, Thrust Rating Indication, and Altitude Alerting. The matrix of failure sensors versus mode availability or reversion represents a matrix of about 2100 separate conditions. Each of these was tested in the validation facility and verified that the correct reversion and annunciation occurred, or that no effect on a given mode was observed and none was intended. The significance of the totality of the simulation facility is evident only if it is noted that every sensor affecting every mode can be realistically tested and the annunciation to the flight crew verified using actual flight hardware operating in a realistic flight environment" (reference 6).

"Similarly, a significant test effort on the validation facility was a verification of the accuracy and comprehensiveness of Failure Modes and Effects Analyses conducted on the system. The FMEA is the primary effort used to scrutinize details of the design in order to assure its overall safety. It examines the failure mechanisms and their consequences in every sensor signal path, computation and control element, and mechanical control or display device. Effectively, it confirms the adequacy of the monitoring and shutdown procedures. With the system operating in the modes examined in the FMEA's postulated, failures were inserted and their effects observed on performance, detection via monitoring, shutdown and annunciation of the failure. The significance of verifying the accuracy of these analyses in a fully operative system environment cannot be overstated" (reference 6).

In the YC-14 program (reference 3) a test and failure identification panel was used to run tests in the aircraft to verify the flight worthy status of both the flight control electronics and the interfacing equipment critical to the flight control electronics operation. "Failure monitoring of input sensor signals and discretes is accomplished in software using cross-channel comparison algorithms. The computers have extensive in-line monitoring and cross-channel data link monitoring. Servos are monitored by comparison of the position feedback signals using cross-channel comparison and techniques similar to those used for sensors. Each monitor identifies the failed element and issues channel isolate requests to redundant hardware isolation logic that is used to reconfigure the system to two-channel operation in the event of failure. The hardware logic allows a channel to isolate itself and also allows the channel to be isolated when the monitors in the other two channels decide that it is failed. Optical data transmission between the channel isolation logic maintains the electrical separation of the channels. Following the isolation of a channel, the signal selectors are reconfigured to select the average of the remaining signals. In the event of a second failure occurring in any of the two remaining active channels, both remaining channels are disengaged" (reference 3).

"The fault monitoring system detected in flight every failure. Required redundancy management functions of system reconfiguration, crew warning, and fault identification operated correctly and allowed the continued safe operation and quick diagnosis of the fault" (reference 3).

The F-18 (reference 2) used a failure modes and effects test sequence similar to that for the DC-9 Super 80, YC-14, and NASA F-8.

9.6.1.3 Stress Test. "This class of verification exposed more anomalies per test than any other. The stress test procedures are listed in table 9-5 for each of the test types. These tests were, for the most part, conducted with the full iron-bird operational, including actuators. Surface commands were monitored directly, and post-test core dumps were examined to establish recovery for fault sequences. The results of these tests are summarized in table 9-6" (reference 1).

TABLE 9-5. STRESS TEST PROCEDURES (Reference 1)

Test Type	Procedure
Timing Overload	Increase sample rate until minor-cycle period equals compute time.
Phantom Job	Insert asynchronous job which "drifts" through minor-cycle period, and also overflow job queues.
Parametric Variations	Reduce timing tolerances and fail-counter values during quiescent and power-transient conditions (during restarts).
Power Faults	Interrupt or reduce prime 28-Vdc power in one or more channels.
Induced Operator Errors	Erroneous operation of mode panels and cockpit controls.

TABLE 9-6. STRESS TEST RESULTS (Reference 1)

Test Type	Result
Timing Overload	. No errors found
Phantom Job	. Error in downlink software exposed . Interference with sync process noted
Parametric Variations	. Minimum acceptance values determined . Margins sufficient
Power Faults	. Exposed major software problems in recovery process . Sync and restart software design errors . Insufficient tolerance on failure counters
Induced Operator Errors	. No system FDIR software errors found . Applications software errors found

"The stress tests exposed problems not apparent in the more controlled and systematic module verification tests and distinct failure modes and effects to produce sequences of operation that were not considered in the design phase or produced in the controlled fault testing.

"The stress tests exposed problems that existed almost exclusively in the restart recovery process. The modifications improved nuisance-fault immunity substantially without materially affecting fault-detection capability" (reference 1).

9.6.1.4 Piloted Failure Modes and Effects Test. "The formal and systematic open-loop failure modes and effects test are generally conducted under static conditions, that is, under conditions which approximate cruise flight. Although many failure effects can be evaluated independently of the environment in which the system is operating, it is not possible to evaluate all failure effects in this manner. In the F-8 DFBW program, a closed-loop piloted FMET series was performed. Table 9-7 lists the type of tests accomplished. These tests were performed in the iron bird in a real-time simulation mode with all systems active. The pilot was aware of the fault to be injected and the time it was inserted. This was done so that the pilot could analyze the fault and select conditions where the consequences might be most severe. In many cases, the same fault was inserted several times to permit the pilot to examine the results under various dynamic conditions.

TABLE 9-7. PILOTED FMET SERIES (Reference 1)

TEST METHOD

- . Induce Single and Multiple Like Faults During Closed-Loop Piloted Simulation on "All-Up" Iron Bird
- . Pilot is aware of fault type and insertion time
- . Pilot/System/Vehicle Response is Analyzed

FAULTS INDUCED

- . Motion Sensors
 - Rate Gyros
 - Accelerometers
 - Attitude
- . Stick/Pedal Transducers
- . Computer/IFU
 - One, Two, Three Channels
- . Trim Fails
 - Open
 - Runway
- . Bus Power
 - One, Two, Three Channels
- . Hydraulic Power
 - PC1, PC2, Utility
- . Actuation
 - Individual Channels
 - Single Surfaces
 - One Horizontal Aileron, One Rudder
 - Both Ailerons
 - Both Ailerons, One Horizontal
- . Total Primary System Failure Without Annunciation
 - Pilot Must Select Bypass System

"This phase of testing is very critical. Faults are coupled with pilot response and the total system performance can be assessed in realistic circumstances. This is important not only in uncovering problem areas, but also, if the design is sound, in giving the pilot a measure of confidence in overall system integrity. This is ultimately a critical step in the flight qualification process.

"The results of these pilot's FMETs are summarized in table 9-8. Problems were identified and corrected, although very few actually occurred. It must be noted here that the pilot had been involved in the development of the system reconfiguration policy and had evaluated interim configurations prior to the formal FMET series.

TABLE 9-8. RESULTS OF PILOTED FMET SERIES (Reference 1)

PROBLEM AREAS

- . Open Failures on Stick/Pedal LVDTs Not Detected Quickly Enough (Bias Added)
- . Some Annunciation of Failures Confusing (Logic Changed)
- . Runaway Rudder Trim Detection Time Too Long (Limit Trim Authority)
- . Change of Stick Trim With Automatic Reconfiguration Requires Pilot Action

OVERALL TEST RESULTS

- . No Loss of Control For Any Survivable Fault Condition
- . Automatic Reconfiguration Did Not Couple With Pilot Response
- . The Airplane Can Be Landed With the Following Surfaces Active:
 - One Horizontal Stabilizer
 - One Aileron or Rudder (Lake-Bed Recovery With No Ailerons)
- . Fail-Operational Results For All First Faults
- . Pilot Not Required to Take Unusual Action For Any Survivable Fault

"The overall test results were significant. For all survivable faults, the pilot was able to maintain control of the aircraft. This was not the case in the Phase I F-8 DFBW program, where a category of faults in the single-channel digital system coupled with normal pilot responses produced some unrecoverable conditions. This was due to detection and reconfiguration processes taking up to one second. No such cases exist in the triplex Phase II DFBW system" (reference 1).

9.7 PILOTED-MISSION-PROFILE TESTING.

"These tests consisted of pilot evaluations of the functional performance of the DFBW control system on the iron-bird during real-time simulation. These tests served several purposes:

- (1) Refinement of control laws for best flying qualities.
- (2) Exposure of functional performance problems due to design or software deficiencies.
- (3) Determination of the degree of system nuisance-fault immunity under normal operating conditions.
- (4) Demonstration of system integrity to the pilot, and, in fact, to the entire project team during conditions closely simulating flight.

"It is not only important in the test phase to demonstrate that the flight-control system can tolerate faults, but that also over a period of months the system can be relied upon to perform its intended functions without problems. This is another critical step in the flight qualification process. A system which fails frequently under normal operating conditions does not produce the confidence that it will perform the intended job when needed most. A nuisance fault is more than a nuisance. In a system like the triplex F-8 DFBW primary flight-control systems, a nuisance fault is indistinguishable from a permanent hardware failure if it causes loss of a channel or a sensor. The integrity of the system can then be thought of as that quality which makes it both reliable and fault tolerant. It is a system that can be depended on. The mission-profile test provides information on the integrity of the system" (reference 1).

9.7.1 Aircraft Integration Tests.

"Final integration tests were accomplished on the F-8C aircraft itself. The tests performed in the aircraft are listed in table 9-9. The tests listed are all straight forward and were accomplished relatively easily. Some problems were identified during this phase in testing:

- (1) Power-transient susceptibility of bypass system during primary-system power cycling.
- (2) Scale-factor discrepancies in the instrumentation system.
- (3) Errors found in preflight test program software in routines not tested on iron-bird.
- (4) Tolerance adjustments required in several preflight program routine.
- (5) Channel failure frequently caused by computer hardware problems.
- (6) Excessive mechanical offset a stick linear variable differential transformer (LVDT).

"EMI testing was composed of two separate test categories. The first involved the superposition of noise on the source 28-V dc power lines with the system operating in the normal manner. The second test involved the operation of all electrical systems on the aircraft during normal operation of the DFBW system. There were no observed anomalies during either test.

"All other test results were positive. With control system loop gains set to four times their maximum value, no adverse structural-mode interactions occurred. The identified problems were corrected and the aircraft was prepared for high-speed taxi tests and first flight" (reference 1).

TABLE 9-9. AIRCRAFT INTEGRATION TESTS (Reference 1)

INSTALLATION TESTS

- Power Connections
- Continuity Checks
- Sensor Sign Verification

END-TO-END TESTS

- Stick-to-Surface Calibrations
- Dynamic Surface Responses

FUNCTIONAL CHECKS

- Moding of Flight-Control System
- Primary/Bypass Transfer Dynamics

RESONANCE TESTS

- Increased Loop Gains to Identify Structural Resonance Problems

PREFLIGHT SELF-TEST CALIBRATION

- Self-Test Tolerances
- Verify Operation/Passability

EMI SUSCEPTIBILITY/RADIATION

- Effect of Aircraft Systems on DFBW System
- Effect of DFBW System on Aircraft Systems

ENGINE RUN

- Functional Tests
- Dry Run of Preflight Test

9.8 SYSTEM FLIGHT TEST.

"For the purposes of this discussion, the engine runs and taxi tests are considered part of the flight-test program. As a matter of policy, Dryden Flight Research Center requires the airplane and systems be flight qualified prior to the taxi test. The flight-test program is the ultimate validation test. Here, the system and airplane demonstrate the mission suitability of the design in the actual flight environment" (reference 1).

9.8.1 High-Speed Taxi Tests.

Prior to the initial flight test, this series of taxi tests are normally performed with the concluding test being a high-speed taxi test. Often these tests reveal some fault that has not been previously detected. In the case of the F-18, "taxi tests conducted the week before first flight indicated sensitivity problems with the nose wheels steering" (reference 2). In the F-8 DFBW taxi test, channel A failed during the shutdown procedure. "The channel failure was found to be due to inadvertent crew operation of one of the built-in fault-injection circuits on the ground-test cart" (reference 1).

9.8.2 Flight Tests.

In the F-18 flight test program, "critical flight parameters including failure states of the FCS components were monitored in real-time during the flight" (reference 2). In the NASA F-8 program, 50 flights were conducted over approximately 55 flight hours.

Table 9-10 summarizes the in-flight computer and interface unit failure experience for the NASA F-8 program. "The computer faults on flights 2 and 3 provided fail-operational experience earlier than expected. The system FDIR response was nearly identical to that observed during ground testing. In both cases, the two remaining channels declared the offending channel hard-failed, and operation continued with no degradation in flying qualities. Precautionary routine landings were made on the remaining two channels. Post-flight analysis of the fault on flight three did disclose a software design error" (reference 1).

9.9 SOFTWARE ANOMALY EXPERIENCE.

In the NASA F-8 program, a number of software anomalies were experienced. These are summarized in table 9-11. "One final software fault experience is germane to this subject. An error in the manufacturer-supplied support software for the computer results in an erroneous load tape for a new software release. This error was not detectable by ordinary means and was only discovered in ground testing because it was in executing code. This error was in an area of the code which had not been modified in the new release and which ordinarily would not have been reverified. This emphasizes the point that software faults can be introduced in a very subtle manner when code is modified after qualification" (reference 1).

TABLE 9-10. IN-FLIGHT COMPUTER AND IFU FAILURE EXPERIENCE (Reference 1)

Flight	Failure	Comments
2	Computer memory fail	<ul style="list-style-type: none">. Memory parity error unrecoverable. Multiple restarts could not restore operation
3	Computer memory fail	<ul style="list-style-type: none">. Channel declared failed by partners. No surface transients. Pilot did not observe performance change. Routine landing on two channels
17	Interface unit I/O (transient fault, failed hard on ground)	<ul style="list-style-type: none">. Restart restored normal operation. No surface transient. Flight continued per plan
19	Computer stopped execution	<ul style="list-style-type: none">. Partners assumed it was off. No restarts. Routine landing on two channels
22	IFU Component	<ul style="list-style-type: none">. Channel failed by partners/self. Routine landing on two channels

TABLE 9-11. SOFTWARE ANOMALY EXPERIENCE (Reference 1)

Software Subsystem	Types of Anomalies	How Found	Comments
System redundancy management	Fault-detection logic	Analysis of flight-computer fault	. Design error . Noncritical
	Deficiencies in fault recovery logic	Analysis of ground-test data	. Could prevent recovery from multiple transient faults
	Sensor fault logic errors	Ground operations	. Minor system effects
Control laws	Restart initialization	Ground operation	. Minor system effects
	Miscellaneous flag handling	Ground operation	. Minor system effects
Input/Output	Incorrect internal interrupt handling	Ground-test data analysis	. Resulted in occasional restarts on ground and one in-flight . Noncritical
Ground-test programs (loader, displays, preflight test)	Nuisance-type faults	Ground operation	. Noncritical . Minor effects

9.9.1 Software Modification.

The rules governing software modification on the F-8 program were:

- (1) All changes must be positively verified and validated on the iron bird.
- (2) All changes must be evaluated by a pilot in real-time simulation on the iron bird.
- (3) A change to the system redundancy management software requires

FMET steps applicable to that subsystem be repeated.

"Software changes involve more risk than the generation of the original software because they are exposed to few hours of operation prior to use. Thus, new software does not receive the implicit verification that often exposes errors" (reference 1).

In the F-18 program, software changes were first made in a core memory flight software program and flown on the flight simulator. Upon satisfactory demonstration, Programmable Read-Only Memory (PROM's) were burned for incorporation in flight units (reference 2).

In the Boeing YC-14 program (reference 3), software-related failures were due primarily to errors in the software requirements.

"(1) Software validation was a difficult task because of the large combination of events that could possibly result in error response.

(2) Software component level testing, open-loop testing, and closed-loop testing were complementary techniques; each had an important role in software validation.

(3) Automation of testing will be essential to meeting schedules while providing the number of test cases required to give sufficient coverage.

(4) Input-to-output, open-loop testing should be performed at the digital interface to preserve accuracy of tests. YC-14 experience showed that analog conversion inaccuracies resulting from testing analog-sensor input to analog-servo-position output were excessive.

(5) System level testing should test to satisfy functional requirements in the normal no-failure state and also confirm that requirements are satisfied in failure state configurations" (reference 3).

9.10 SYNOPSIS OF CURRENT DIGITAL VALIDATION EXPERIENCE.

"We believe the primary validation method must exercise the actual system, including as many of the real peripheral devices as possible, with simulated stimulus inputs that are as realistic as practical. Every possible fault that can be imagined should be considered, and where possible, induced during system tests. In order to effectively induce faults, a system must have been designed from the beginning to allow fault introduction. A system must be stressed to the maximum extent possible. The philosophy of testing should not be to show that the system works properly but to make it fail.

"This type of testing was the primary validation method used on the F-8 DFBW system. Even more detailed tests are recommended for future systems; one example is the introduction of transient faults. In the F-8 DFBW system test, some of the simulator hardware failures were injected as 'hard failures' only. Furthermore, no attempt was made to synchronize these failures within the control cycle. An expanded test would be made more detailed by

(1) Simulating failures as both 'hard' and 'transient'.

(2) Vary the length and period of transient failures.

(3) Control and vary the relationship of the failure to the control cycle of the system" (reference 1).

9.11 REFERENCES

1. Szalai, Kenneth, J., Jarvis, Calvin R., Krier, Gary E., Megna, Vincent A., Brock, Larry D., and O'Donnell, Robert N., Digital Fly-By-Wire Flight Control Validation Experience, R-1164, The Charles Stark Draper Laboratory, Incorporated, June 1978.
2. Kisslinger, R. L., Momeno, W. J., and Urnes, J. M., Design and Development of the Control Laws for the F-18 Primary Flight Control System, Pages 158-165, NAECON 1980.
3. Martin, D. L., YC-14 Digital Flight Control Experience, Pages 166-173, NAECON 1980.
4. Lockenour, Jerry L., Saworotnow, Ivan, Kesler, Don F., Digital Flight Control - The Generation of the 80's, Pages 182-190, NAECON 1980.
5. Spradlin, Richard E., The 757/767 Flight Management System Laboratory Test Program, 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.
6. Urling, Herbert R., DC-9 Super 80 Digital Flight Guidance System Simulation Techniques for Certification, Pages 57-79. Proceedings of the 1981 RTCA Assembly, Radio Technical Commission for Aeronautics.
7. Erwood, R. G., McCorkle, R. D., Parente, J. J., et al, Digital Flight Control Redundancy Management System Development Program, AFFDL-TR-79-3050, Vol. II, Boeing Aerospace Company, May 1979.
8. Godoy, S. G., and Engels, G. J., Sneak Circuit and Software Sneak Analysis, Journal of Aircraft, Vol. 15, No. 8, August 1978, Pages 509-513.
9. Elson, Benjamin M., Research into Equipment Malfunctions Intensifies, Aviation Week and Space Technology, October 27, 1980.
10. Environmental Conditions and Test Procedures for Airborne Equipment, Document No. RTCA/DO-160A, Radio Technical Commission for Aeronautics, January 1980.
11. Hall, Preston S., Vibration Test Level Criteria for Aircraft Equipment, AFWAL-TR-80-3119, Flight Dynamics Laboratory, Wright-Patterson AFB, OH.
12. McGough, J. G., Use of Emulation in Fault Analyses of Digital Systems, Pages 29-42, Proceeding of the 1981 RTCA.
13. Gill, Chris, and Thompson, John A., A Unified Test Harness System for Avionics Software Development, Pages 553-560, NAECON 1980.
14. Patterson, Alton E., Air Force Integration Support Facilities; Their Total Utility, Pages 126-135, NAECON 1980.

15. Corder, David R., Oklahoma City Air Logistics Center Approach to Embedded Computer System Support, Pages 143-147, NAECON 1980.
16. Flight Simulation/Guidance Systems Simulation, AGARD CP-198, North Atlantic Treaty Organization, 1975.
17. Klos, L. C., and Straeter, T. A., Applying Modern Software Engineering Tools to Avionics Systems Development, AIAA Paper 81-2259, Pages 212-221, 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.

SECTION TEN
RECOMMENDED VALIDATION PROCEDURES

TABLE OF CONTENTS

	Page
SECTION 10	10-1
10. RECOMMENDED VALIDATION PROCEDURES	10-1
10.1 Overview	10-1
10.2 Advanced Digital Integrated Flight Control and Avionics Validation Methodology	10-1
10-2.1 Concept	10-10
10-2.2 System Definition Phase Activities	10-10
10-2.3 System Design Phase Activities	10-18
10-2.4 System Full-Scale Development Activities	10-24
10-2.5 System Integration/Test Activities	10-25
10-2.6 Production and Deployment Activities	10-28
10-2.7 Operation and Maintenance Activities	10-28
10.3 References	10-30

LIST OF ILLUSTRATIONS

Figure		Page
10-1	Systems Design/Development Process (Industry/FAA)	10-2
10-2	Expansion of FAA Interest/Viewpoint	10-2
10-3	Software Life Cycle Activities Time Relationship	10-3
10-4	Digital Systems Validation Activities Sequence	10-4

LIST OF TABLES

Table		Page
10-1	Activities and Documentation	10-7
10-2	Seven Steps of Systems Engineering (Reference 1)	10-10

SECTION 10

10. RECOMMENDED VALIDATION PROCEDURES

10.1 OVERVIEW.

This section presents a recommended methodology for verification, validation, and maintenance of advanced digital integrated flight controls and avionics systems. The roles and responsibilities of the various agents (end customers, developers/manufacturers, independent test organization (ITO), regulatory agency) are described. Specific analysis and design aids (models and tools) are discussed along with the times in the system life cycle during which these analysis and design aids are used. The reason that this section encompasses all of the foregoing activities is illustrated in figures 10-1 and 10-2. While industry is performing all the work which must be completed prior to the final validation steps, there needs to be a great deal of interaction between the regulatory agency and the industry developer throughout these early phases in the system life cycle in order to assure that all things needed for certification have been satisfactorily accomplished.

In order to better illustrate the interaction in each phase of the system life cycle, the activities illustrated in figures 10-1 and 10-2, require additional definition. Figure 10-3 depicts the overlap of activities in each phase of the software portion of the system life cycle. A flow chart of activities, such as a PERT chart, does not clearly illustrate the probable overlap of activities at the completion of one phase and the beginning of another. Figure 10-4 is a flow chart of this type which depicts the additional activities, discussed in the following pages, that must be accomplished to validate the digital system's hardware and software. These activities could be further subdivided into smaller discrete tasks, but this will not be done in this handbook. The reader should keep in mind that, while the figure 10-4 activities imply a time relationship, there may be a slight overlap between the completion and start of sequential tasks.

10.2 ADVANCED DIGITAL INTEGRATED FLIGHT CONTROL AND AVIONICS VALIDATION METHODOLOGY.

Figure 10-4 depicts many of the major activities that must take place during the system definition, design, full-scale development, operational test and evaluation, airworthiness/certification, production/deployment, and operations/maintenance phases of the system life cycle. The figure depicts the major activities and their time relationship and provides supplemental information for each activity. This information includes, as shown in the chart legend (table 10-1), the agent responsible for that activity, the number and title of the activity which correlates with later paragraphs in this section, and the documentation products input to and output from that activity (table 10-1). In addition, major decision points are indicated by the diamond shaped logical decision box. The chart also depicts the logical relationship of the activities in terms of the AND and OR conditions relating the previous and subsequent activities. For example, the first activity depicted occurs during the concept formulation phase and indicates that the manufacturer is the responsible agent. The activity number, 0.1, indicates that this is the first activity of the concept formulation phase on the chart. Note that the input product to the activity is a system analysis report. After completion of this activity, activities 0.2, 1.1, 1.2, and 1.3 all require that activity 0.1 take place. Further note that for activity 1.4 to begin, activities 1.1 and 1.2 must be complete. The logical conditions, OR and AND, obey the same truth table relationships that they do in computer science.

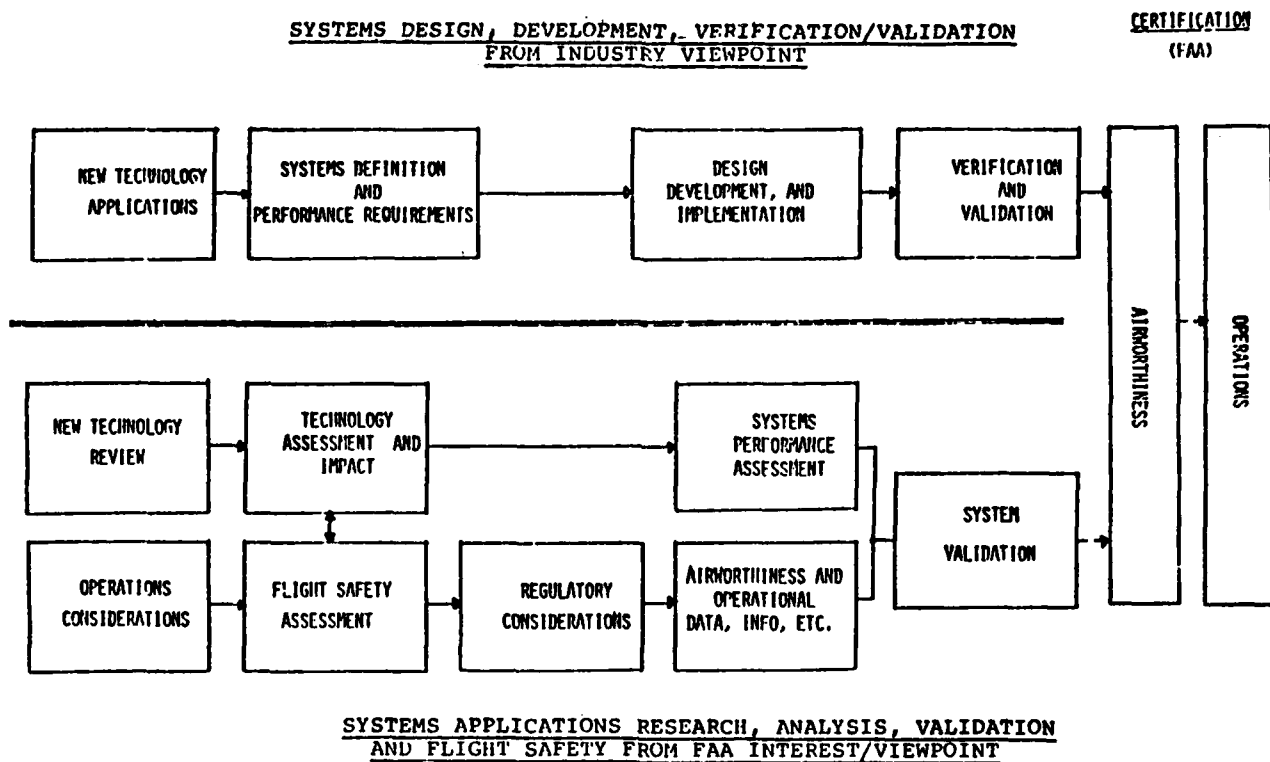


FIGURE 10-1. SYSTEMS DESIGN/DEVELOPMENT PROCESS (INDUSTRY/FAA)

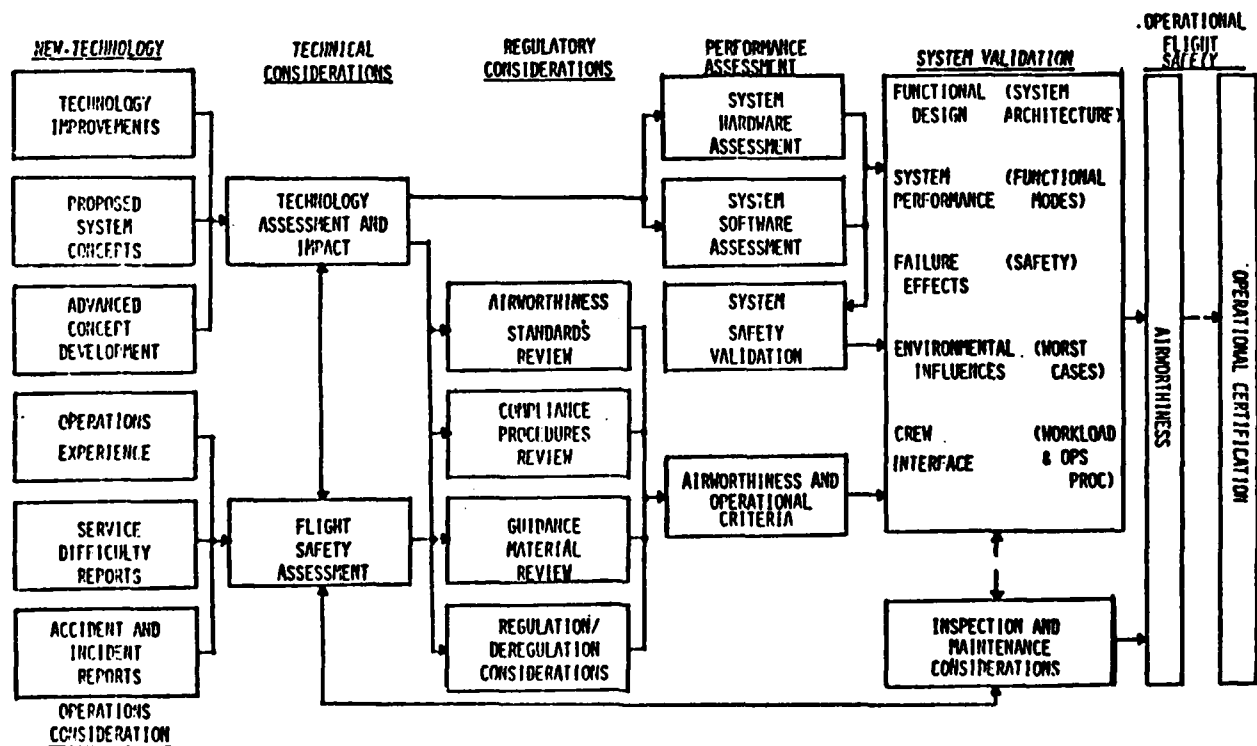


FIGURE 10-2. EXPANSION OF FAA INTEREST/VIEWPOINT

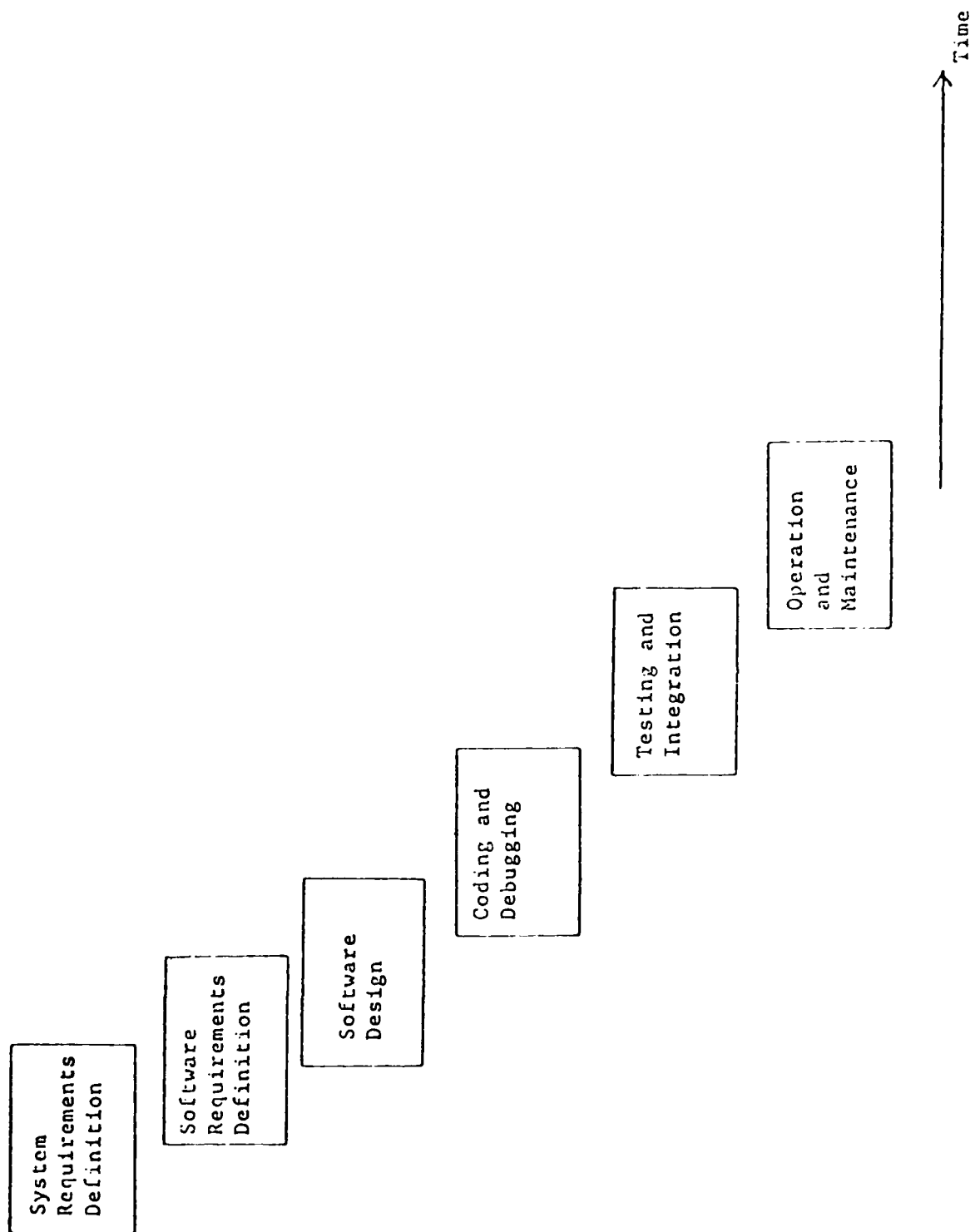
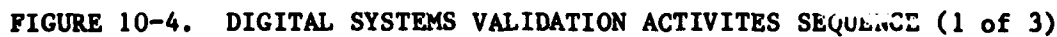


FIGURE 10-3. SOFTWARE LIFE CYCLE ACTIVITIES TIME RELATIONSHIP



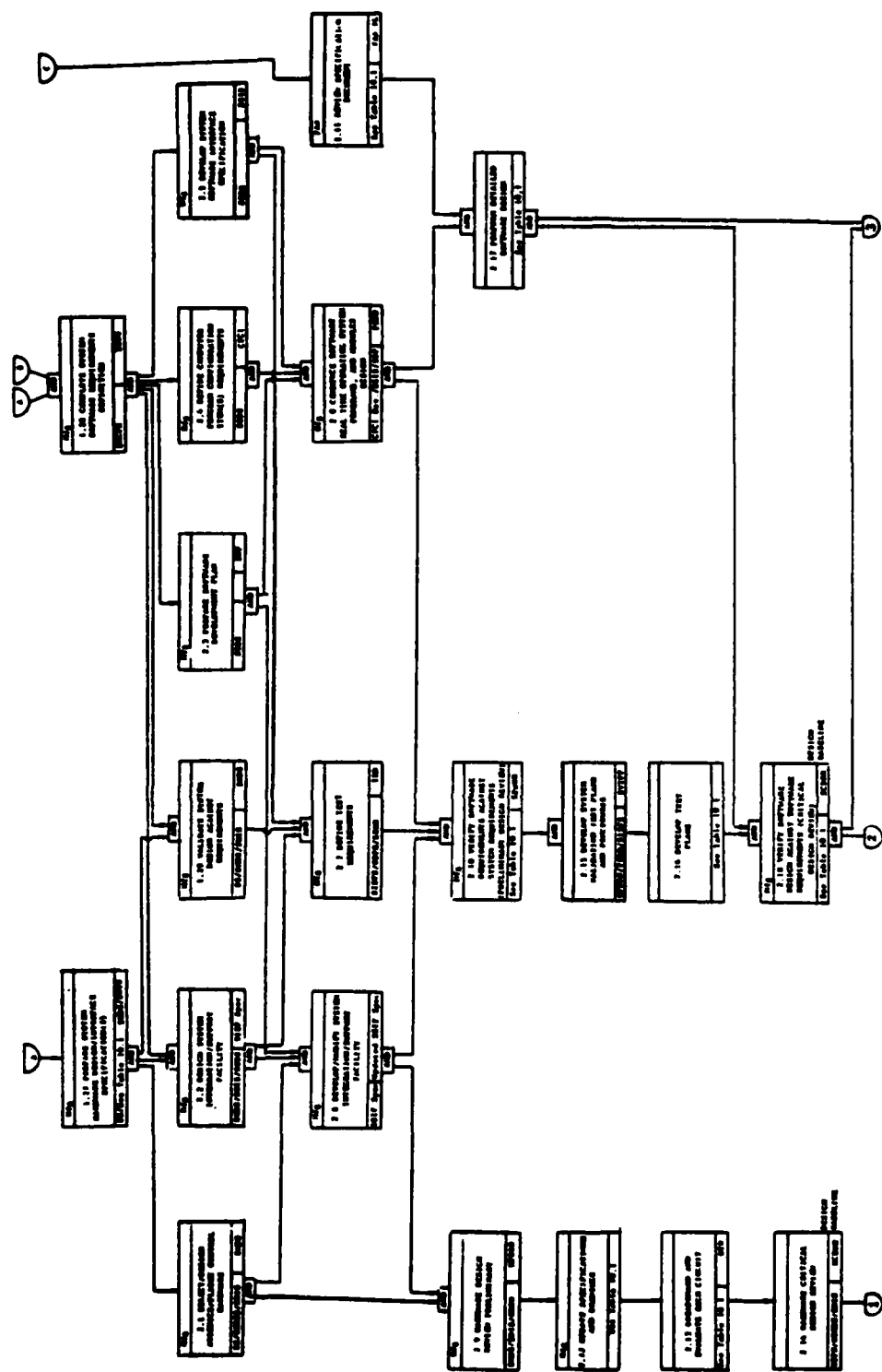


FIGURE 10-4. DIGITAL SYSTEMS VALIDATION ACTIVITIES SEQUENCE (2 of 3)



TABLE 10-1. ACTIVITIES AND DOCUMENTATION

Activity	Documents	
	Input	Output
0.1 Plan to Develop New System		System Analysis (SA) Report
0.2 Prepare Program Management Plan		Program Management Plan
1.1 Define Mission Performance Requirements/Operational Environment	System Analysis Report Federal Aviation Regulations	System Requirements (SR) Report
1.2 Define Vehicle Characteristics and Parameters	System Analysis Report Federal Aviation Regulations	Vehicle Characteristics Report
1.3 Define Method to be Used to Evaluate Reliability/Safety	FAA Advisory Circular 25.1309-XX	Reliability/Safety Assessment (RSA) Plan
1.4 Define Required Systems Functions for Each Flight Phase	System Requirements Report	System Functions Report
1.5 Determine System Functions Criticality Category	FAA AC 25.1309-XX System Functions Report	Functions Criticality (FC) Report
1.6 Define Baseline Modes Performance Required for Each Function	Functions Criticality Report	Baseline Performance Requirements (BPR) Report
1.7 Define Quantitative Mission Reliability Goals	Functions Criticality and System Requirements Reports	Mission Reliability Goals Report
1.8 Define Classes of Faults/Events to be Tolerated	Functions Criticality Report	Fault Tolerance Requirements Report
1.9 Finalize System Requirements	Baseline Performance Requirements, Mission Reliability Goals,...Reports	System Specification
1.10 Define Candidate System Architectures	Fault Tolerance Requirements Report	System Candidate Architecture (CSA) Report
1.11 Develop/Adopt Configuration Management Plan	Program Management Plan and System Specification	Configuration Management Plan
1.12 Develop Documentation Tree/Configuration Item Index	Configuration Management Plan and System Specification	Configuration Item Index
1.13 Select Fault Detection Algorithms	Fault Tolerance Requirements Report and CSA Report	Fault Detection Algorithms (FDA) Report
1.14 Devise Recovery Procedures	FDA Report	Recovery Procedures (RP) Report
1.15 Develop Baseline Modes Switching Logic for Each Function	BPR, CSA, FDA, and Recovery Procedure Report	Mode Switching Logic (MSL) Report
1.16 System Requirements Review (SRR)	System Specification	SRR Report
1.17 Define Baseline Functional Requirements in Terms of Hierarchical Modes	BPR and MSL Reports	Mode Hierarchy (MH) Report
1.18 Obtain Candidate Subsystems and Components Data	CSA and Mission Reliability Goals Report	Subsystems and Components Data (SCD) Report
1.19 Perform Reliability Analysis	Mode, Hierarchy and SCD Reports Reliability/Safety Analysis Plan	Reliability Assessment (RA) Report
1.20 Evaluate Performance	RA, Mode Hierarchy, SCD, MSL, RP, FDA, SA, SR and CSA Reports	Performance Evaluation (PE) Report
1.21 Conduct System Safety Assessment	PE, RA, SR, FC, SCD, MSL, RP, FDA	System Safety Assessment (SSA) Report
1.22 Establish Initial Man/Machine Task Allocation	PE, RA, SCD, MH, MSL, FDA Reports	Workload Assessment Criteria (WAC) Report
1.23 Partition/Allocate Functions to Software/Hardware	WAC, PE, MH, MSL, RP, FDA Reports	Function Allocation (FA) Report
1.24 Establish Initial System Operating/Control Procedures	WAC, FA Reports	System Operating/Control Procedures (SOCP) Report
1.25 Review Reliability Analysis	RA Report	FAA Review Letter

TABLE 10-1. ACTIVITIES AND DOCUMENTATION (Continued)

Activity	Documents	
	Input	Output
1.26 Review System Safety Assessment Analysis	SSA Report	FAA Review Letter
1.27 Prepare System Hardware Design/Interface Specification(s)	System Spec. and all Preceding Activities Reports	System Hardware Development (SHD) Spec(s) System Hardware Interface (SHI) Spec.
1.28 Complete System Software Requirements Definition	SOCF	System Software Development (SSD) Spec.
1.29 Validate System Design Against Requirements	System, SHDS, SHI, SSD Specifications	System Requirements Review Report
2.1 Select/Design Avionics/Flight Control Hardware	System, SHDS, SHI Specification	System Hardware Design Specification(s)
2.2 Design System Integration/Support Facility	SHDS, SHI, SSD Specification	System Integration/Support Facility (SISF) Specification
2.3 Prepare Software Development Plan	SSD Specification	Software Development Plan (SDP)
2.4 Define Computer Program Configuration Item(s) Requirements	SSD Specification	CPCI Development Specifications
2.5 Develop System Software Interface Specification	SSD Specification	System Software Interface Specification
2.6 Develop/Modify System Integration/Support Facility	SISF Specification	Updated SISF Specifications
2.7 Define Test Requirements	SISF Spec., SDP, SRRR	Test Requirements Description
2.8 Commence Software Real Time Operating System, Programs, and Modules Design	CPCIs Dev. and SSI Spec. SDP	CPCI Preliminary Software Design Document(s) (PSDD)
2.9 Hardware Preliminary Design Review (PDR)	SHD and SHI Spec.; Hardware Design Description Report	Hardware Preliminary Design Review (PDR) Report
2.10 Verify Software Requirements Against System Requirements (Software PDR)	CPCI Preliminary Software Design Documents; CPCI Development and SSD Spec.	Software PDR Report
2.11 Review Specifications	System SSD, CPCI, SHD, SHI Specifications	FAA Review Letter
2.12 Update Specifications and Drawings	Hardware PDR Report SHD and SHI Specifications	Updated Specifications and Drawings
2.13 Breadboard and Evaluate Each Circuit	Updated Specifications/Drawings Hardware Test Plans	Hardware Test Reports
2.14 Hardware Critical Design Review (CDR)	Hardware Test Plan and Reports; Updated SHD, SHI, Specifications	Hardware CDR Report
2.15 Develop System Validation Test Plan and Procedures	Software PDR Report, PSDD, SSF Specifications	System Validation Test Plan and Procedures (SVTPP)
2.16 Develop Test Plans	SVTPP, SISF Specifications, PSDD, Software PDR Report	Stand Alone Test (SAT) Module Integration Test (MIT), System Integration Test (SIT) Plans
2.17 Perform Detailed Software Design	SSD, CPCI Specifications; CPCI PSDD	Draft CPCI Product Specifications
2.18 Verify Software Design Against Software Requirements (Critical Design Review)	Draft CPCI Product Specification Test Plans/Procedures	Software CDR Report
3.1 Build/Test Prototype Hardware Subsystems	Hardware CDR and Test Reports Hardware Test Plans	Subsystem Test Report

TABLE 10-1. ACTIVITIES AND DOCUMENTATION (Continued)

Activity	Documents	
	Input	Output
3.2 Integrate/Test Hardware System's Subsystems	Hardware Integration Test Plan (HITP)	Hardware Integration Test Report (HITR)
3.3 Code Modules and Debug	Draft CPCI Product Specification CPCI PSDD	Code/Debug Status Report
3.4 Perform Stand-Alone Module Testing	Stand-Alone Test Plan	Stand-Alone Test Report
3.5 Verify Code Versus Design	SAT Plan and Code	SAT Code Validation Report
3.6 Test Modules	SAT Plan and Coded/Assembled Modules	SAT Module Validation Report
3.7 Integrate Modules and Test Each CPCI	MIT Plan and Validated Modules	Module Integration Test Report (MITR)
3.8 Test Module Integration/CPCI	MIT Plan and Integrated Modules	MIT Validation Report (MITVR)
4.1 Integrate System (Software and Hardware) and Test in Lab	SDP, Hardware Integration Test Report, MITR	System Integration Report (SIR)
4.2 Test System Integration	MITR, SIR, MITVR	System Integration Validation Report (SIVR)
4.3 Integrate System Into Aircraft	SIR	Aircraft Integration Report
4.4 Perform System Validation Test	SIR, SIVR, MITVR	System Validation Test Report (SVTR)
4.5 Perform Flight Tests	Flight Test Plan (FTP)	Flight Test Report (FTR)
4.6 Perform Acceptance Tests	Acceptance Test Plan	Acceptance Test Report
4.7 System Certification	System Certification Plan	System Certification Report
4.8 Perform Operational Test and Evaluation	OT&E Plan	OT&E Report
5.1 Production and Deployment	Warranty Plan (WP), Spares Provisioning Plan (SPP)	Warranty Action Reports (WAR)
6.1 Operation and Maintenance	Maintenance Plan, Maintenance Training Plan/Material, Systems Integration Support Facility Plan	Maintenance Reports
6.2 Recertification Criteria	FAA System Simulation Test Report; User Maintenance Reports	Recertification Criteria Report
6.3 System Revisions and Implementation of ECN/FAA AD	Maintenance Reports; FAA AD	System Revision Report
6.4 User Modifications	System Revision Report Maintenance Report	User Modification Report

The following subsections of this section describe for each of the major phases of the system life cycle those activities depicted in figure 10-4. Table 10-1 lists the documentation input to or output from each activity.

10.2.1 Concept.

0.1 Plan to Develop New System. Prior to developing a new avionics and flight control system, the manufacturer will have determined that there is a need and a market for the system. During the concept formulation phase, a general plan of approach will be developed. The seven steps of system engineering shown in table 10-2 are applied repeatedly in each phase of the system life cycle. One of the principal outputs of the analysis performed during the conceptual phase is a system analysis report which is required to develop the system requirements fully. This system analysis report presents the results of a comprehensive functional analysis of the system's elements (personnel, equipment, facilities, and support).

TABLE 10-2. SEVEN STEPS OF SYSTEMS ENGINEERING
(Reference 1)

1. Problem Definition
2. Value System Design
3. System Synthesis
4. System Analysis
5. Optimization of Each Alternative
6. Decisionmaking
7. Planning for Action

0.2 Prepare Program Management Plan. Prior to commencing the system definition phase, the developer of the system will normally prepare a program management plan. The program management plan describes the procedures which will be used to control the work during the subsequent phases of the system life cycle. It includes definition of both technical and financial management tools and describes the reporting procedures in detail. The accounting procedures are defined as well as the contracting procedures. The program-management methods such as project-control tools, required backup staff, and line-management structure are defined. The output of this is the program management plan which is used in subsequent phases.

10.2.2 System Definition Phase Activities.

1.1 Define Mission Performance Requirements/Operational Environment. This activity is one of the first that occurs in the problem definition phase. The work that takes place in this activity is critical since the performance requirements defined form the basis for the subsequent design and evaluation of candidate systems against these requirements. The care taken in this step will impact all subsequent activities.

Section 4 of this handbook previously discussed basic mission-related factors and operational environment considerations. As discussed in Section 3, the use of formal systems requirements engineering methodology may be advantageous. Tools such as the Design Analysis System (DAS) (reference 3), Systematic Activity Modeling Method (SAMM) (reference 4), and others described in reference 5 may be of advantage in establishing the requirements of a large system comprising many

subsystems. The output of this task is a compilation of the mission performance requirements and operational environment description and is characterized as a system requirements report.

1.2 Define Vehicle Characteristics/Parameters. This activity requires careful definition of the vehicle characteristics including mass properties as a function of time, aerodynamic data, propulsion system data, and data on all systems the avionics and flight control interface with including:

- (1) Aircraft electrical system
- (2) Flight control surface actuators
- (3) Air data sensors.

Additional data which may be needed include aircraft performance data and specific fuel consumption parameters. The avionics and flight control developer will require more detailed data at subsequent stages that will permit a complete determination of the functions needed for each phase. The output of this activity is a document containing the data describing the vehicle's characteristics and parameters.

1.3 Define Method to be Used to Evaluate Reliability/Safety. This activity may be considered a joint responsibility of the regulatory agency (FAA) and the manufacturer. The FAA recently provided in Advisory Circular 25.1309-1 (reference 6) direction to use a fault tree. As previously discussed in Section 8, NASA and the FAA have been sponsoring the development of computer implemented reliability analysis models such as CARE III (reference 7). It is possible that the developer of the system and the FAA will agree to use one of the computer-aided reliability analysis models such as those discussed in appendix B of this handbook. In addition, appendix B contains a discussion of fault trees and the failure modes and affects analysis techniques. The output of this activity should be a written document which contains a description of the agreed to reliability/safety assessment analysis methodology to be applied to the system under consideration. This is referred to in table 10-1 as the reliability/safety assessment plan.

1.4 Define Required System Functions for Each Flight Phase. This activity requires the application of the methodology of functional decomposition of the overall mission requirements into specific system functions required for each flight phase. Formal methodologies such as those discussed in Section 3 may be used. If a formal methodology is used, it should provide the capability to "support a flexible approach and functional decomposition which may be successfully embellished to accommodate functional, performance, and operational requirements. In other words, this specification and requirements may best be envisaged as the accumulation of a data base which will be progressively updated with the information regarding functional, performance, and operational requirements" (reference 8). The output of this task is a report listing all functions required for each flight phase of the mission and all environmental conditions.

1.5 Determine System Functions Criticality Category. Reference 6 requires that an analysis conducted "using service experience, engineering, or operational judgment, or by using a top-down deductive qualitative analysis" examine each function performed by the system and determine the criticality of each system function; i.e., either nonessential, essential, or critical as previously defined in Section 3, table 3-1. Reference 6 further provides examples of systems which perform critical functions. It further states "the analysis should be clearly

AD-A133 222

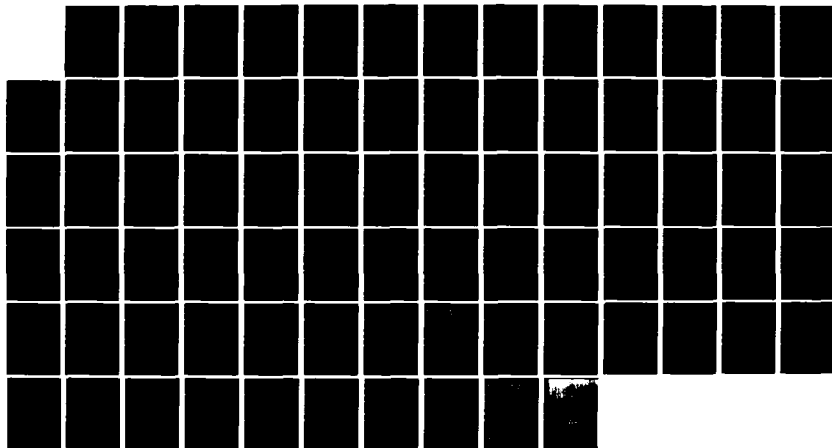
VALIDATION OF DIGITAL SYSTEMS IN AVIONICS AND FLIGHT
CONTROL APPLICATIONS. (U) BATTELLE COLUMBUS LABS OH
E F HITT ET AL. JUL 83 DOT/FAA/CT-82/115
DTFA03-81-C-00059

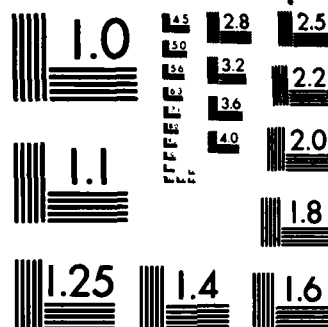
5/5

UNCLASSIFIED

F/G 1/3

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

documented. All the assumptions, sources of reliability data, failure rates, system function type (critical, nonessential, essential), etc., should be concisely documented for ease of reviews. To the extent feasible, the analysis should be self-contained" (reference 6). The output of this activity is a report documenting the criticality category of all system functions.

1.6 Define Baseline Modes Performance Required for Each Function. In some cases, the performance requirements are specified by the regulatory agencies, such as the FAA and CAA as discussed in Section 4 for functions such as autoland in Category I and Category 2 approaches. In other cases, the developer of the system will have to establish the accuracy, data channel capacity, computer throughput, etc., for the baseline mode for each function for each flight phase. Since the baseline mode for all functions should provide the highest overall performance of the system, it is important that all functions which simultaneously require the use of certain resources be included in the analysis. At this stage in the system definition, it is probable that subsequent design changes may necessitate changing some parameters. Therefore, provision must be made for growth in terms of the demand on resources, and data throughput. The results of this activity should be a report documenting the baseline modes performance required for each function as well as the simultaneous demand on resources of the system by the functions.

1.7 Define Quantitative Mission Reliability Goals. The FAA assigned, in reference 6, probability classifications for each of the function criticality categories. The system developer must translate these requirements into quantitative mission reliability goals which can then be allocated to the components of the system. This reliability apportionment is often done using the "similar familiar system's reliability apportionment approach" or the "factors of influence method" (reference 9). As stated in this reference, "both the familiar system factors and the influence methods have their weaknesses when they are used individually. However, combining the two methods produces better results because data are used from the similar subsystems as well as when new subsystems are designed under different factors of influence." The output of this activity is a report which contains the mission reliability goals and the apportionment to each of the system functions.

1.8 Define Classes of Faults/Events to be Tolerated. As previously discussed in Section 5, a fault tolerant system designed to tolerate all known fault types, however small the probability that a fault of a specific type might occur, could be prohibitively expensive. The developer must, based upon the system function criticality category define the classes of faults and events which the system is to tolerate. This must be defined in sufficient detail to permit selection of fault detection algorithms and development of recovery strategies in subsequent activities. The result of this activity is documented in a report.

1.9 Finalize System Requirements. This activity consists of finalizing the system requirements based upon the previous activity and writing the system specification. The system specification must clearly state all system level requirements for each function, its criticality, performance, and the apportioned reliability goals.

1.10 Define Candidate System Architecture. Section 5 of this handbook describes various system architectures. The developer must find one or more candidate system architectures which will form the basis of the design of the system. This architecture must be shown in the subsequent validation step to satisfy the system requirements and specifically the reliability and safety requirements. The result of this activity is a report.

1.11 Develop/Adopt Configuration Management Plan. The manufacturer must either develop or adopt a configuration management plan which will be the basis for the management of the hardware and software configuration throughout the life cycle of the system. Section 11 of this handbook discusses configuration management in general terms. In many cases, corporations already have in place specific configuration management practices. These practices should be compatible with the requirements for software configuration management given in reference 10. The validation activities will make use of the configuration management plan during the validation activities.

1.12 Develop Documentation Tree/Configuration Item Index. A documentation tree should be developed and a method adopted for identifying all documents and the current version of the document. The procedure for assigning configuration item index labels should be explained in the configuration item index document. Reference 10 contains an example of a part number buildup for both hardware and software. If the manufacturer does not have in place a configuration item index, he may wish to follow the recommendations of reference 10. The output of this activity is a documentation tree and configuration item index document.

1.13 Select Fault Detection Algorithms. The fault detection algorithms selection should result in the selection of algorithms that will be implemented to assure:

- (1) Correct operation of each processing unit
- (2) Valid transmission of data between digital subsystems
- (3) Data validity prior to use and subsequent computation.

Various techniques for each of these levels of fault detection were discussed in Section 5 of this handbook. The specific algorithms selected should be documented in a report for subsequent use in the hardware and software specifications.

1.14 Devise Recovery Procedures. The basic recovery procedures possible after isolating the fault are somewhat dependent upon the type of fault. If the fault involves the correct operation of each processing unit's memory, the self-checking circuits and error-correcting codes (dependent upon the code selected) have the ability to detect and correct single bit and detect double bit errors. These error-correcting codes in effect assure that the computer produces the correct outputs, in spite of the fault.

In the case of data transmission errors, recovery techniques may involve a retransmission of the message, with switch to a redundant bus if communication between the digital subsystems is not established within the design number of retries or retransmissions. At a system level, the recovery techniques may involve reloading and restarting of programs as well as complete reconfiguration of the system. The reconfiguration may include the assignment of tasks to processors with the specific task assigned dependent upon the mission phases remaining. Note, that this approach may conflict with the absolute code address for modules discussed in reference 10.

The output of this task is a report documenting the details of each of the recovery procedures. This report is subsequently used in formulating the software and hardware design and interface specifications.

1.15 Develop Baseline Modes Switching Logic for Each Function. The baseline modes switching logic implements the switching between modes as a function of

performance measures not meeting that required for the function or failure of sensors, computers, or data transmission paths. The results of this task should be documented in a report for subsequent use.

1.16 System Requirements Review. This is the first formal review at which the agency responsible for validation normally participates. Often this does not occur and the systems requirements reviewed are limited to the developer and his customer. This system requirement review is generally conducted as a "walk-through" in which a careful review of the system's specification is made against the system requirements with the objective of identifying any deficiencies in the system's specification. These deficiencies must be corrected prior to proceeding with the further definition and design of the system. Additional information on reviews and walk-throughs was presented in Section 8. At the completion of this task, the functional baseline of the system has been established. The output of this task is the system requirements review report.

1.17 Define Baseline Functional Requirements in Terms of Hierarchical Modes. The previously defined baseline modes and mode switching logic for each function must be structured in a hierarchy of modes. Normally, the priority of modes is ordered on the basis that the mode with the best performance is preferred. Depending upon the design approach taken, the system may or may not operate in a mode with the best performance. Generally, the switching between modes for an automatic subfunction is performed by the determination of the availability of the subsystems required for each mode. If all subsystems or all modes are available, the first mode is used. When a failure is detected, system functions will be searched to determine the effect of the failure mode. The system then will revert to the backup mode with the next best performance.

In the case of crew selection of operating mode, it is possible that the crew may manually select a mode having a lower level performance than that which would be automatically selected. The hierarchy of modes must permit this crew selection of any mode with a reversion to the automatic mode selection when the crew so elects.

In developing the hierarchy of modes, rules for functional decomposition should be used (reference 11). This decomposes the system using one of the following decomposition types:

- (1) Sequence (or functional composition) in which the module is decomposed into a series of modules all of which are executed in sequence to accomplish function
- (2) Selector (set partition) in which the module is decomposed into a series of modules, only one of which will be executed
- (3) Coordinator (class partition) to which the module is decomposed into a series of functions; each subfunction acts upon a subset of the input variables to produce a subset of the output variables.

The output of this activity is a report listing the defined baseline functional requirements in terms of the hierarchical modes and the logic and data flow associated with that hierarchy of modes.

1.18 Obtain Candidate Subsystems and Components Data. This activity consists of data collection for use in subsequent reliability and performance analysis as well as design of the system. The output of this activity is documentation of the data to be used in subsequent analyses and design phases.

1.19 Perform Reliability Analysis. An analysis of the theoretical reliability of the candidate system architectures for the various combinations of candidate subsystems and components should be performed and documented in a report. The analysis should be performed using the previously agreed upon methodology. The report should compare the results of the analysis with the previously documented mission reliability goals.

1.20 Evaluate Performance. The performance evaluation will make use of simulation and modeling. For the digital system, a model is built using a simulation language that describes the basic configuration of the system. Performance characteristics can then be determined from the characteristics of the allocated components. The model used in this simulation should reflect not only the system and its components but also the environment the system will operate in.

The simulation model may be driven either by generating input data (probabilistic or Monte-Carlo simulation) or by feeding some representative input data (deterministic or trace-driven simulation) and then simulating the actual behavior of the system. "By repeating this process for various alternative system configurations and parameters, and comparing their performances, one may identify optimal systems structure. Therefore, simulation is essentially the technique of conducting sampling experiments on the model of the system" (reference 12).

"Simulation is often required to provide information on the impact of system throughout, potential queuing problems, and possible overloads of the proposed system configuration. the key issue is to determine the performance of the system prior to its implementation. Performance evaluation seeks to determine:

- (1) Performance characteristics due to algorithmic design,
 - (2) Performance characteristics due to system allocation and configuration, and
 - (3) Performance characteristics due to structure and interfaces"
- (reference 13).

No matter whether the simulation is a self-drive simulation using a statistical description of inputs or a trace-drive deterministic simulation, the results must be properly analyzed to evaluate the performance for use in subsequent design refinements.

In addition to the performance evaluation of the digital system, the performance evaluation using either analytical models or simulation should be done considering the complete closed loop response and accuracy of the system including sensor errors, data transmission delay times, computation delay times, and actuator characteristics to arrive at performance estimates of the "loop system." If a hierarchical structure is used in the development of the evaluation, it is possible that combinations of analytical and simulation models may be used (reference 14).

The output of this activity may be one or more reports evaluating not only the overall performance of the system but the contributions to the performance by the various subsystems. The report(s) document the results of the performance evaluation for each of the alternative configurations under consideration.

1.21 Conduct System Safety Assessment. Using the previously agreed upon system safety assessment methodology, the developer should conduct the system safety assessment for the candidate systems. Results of the assessment should be documented in a report.

1.22 Establish Initial Man/Machine Task Allocation. Prior to initiating this activity, the overall system design may be iteratively improved until a preferred design concept that satisfies the safety, reliability, and performance goals is defined. This activity assumes that the basic concept meets these requirements. The man-machine task allocation is based upon workload and other factors discussed in Section 6 of this handbook. Prior to the application of some of the more sophisticated task allocation tools, functional sequence diagrams, operational sequence diagrams, and action sequence diagrams (reference 15) should have been developed. Reference 15 states "the allocation of functions is a trial and error type of process that proceeds according to the skill of the analyst."

"The allocation process assigns functions and tasks to subsystems that generally reflect techniques of implementation and operation. Obviously, at the highest level of choice, the possible alternatives are:

- (1) Hardware
- (2) Firmware
- (3) Software, or
- (4) Manual (human).

Time was when this decision was easy — software did everything that hardware could not do and the human did not want to. Software was touted as a flexible medium that could compensate for all the inadequacies of hardware of the inefficiencies of the human being.

"The human mind is still the only tool known to be capable of logical analytical thinking — of abstracting conclusions when given a basis of information on which to work. The tendency in some systems has been to try to over automate — to do for the human what he is more capable of doing for himself. The goal of any system, in its ultimate use by and for humans, is to present the information required for human judgment in a manner that best aids decisions. Therefore, the goal is to allocate to the data processing systems those functions which a human cannot (because of time constraint) or will not (because of tedium) do for himself, and allocate to the human those functions that involve value judgments" (reference 15).

Whatever method is used to allocate the task between the human and the machine, the result must be documented in a report.

1.23 Partition/Allocate Functions to Software/Hardware. "The concept of 'software first' is reaching acceptance. Thus, the software is designed through at least an intermediate design stage prior to determining hardware requirements, and software requirements are driving hardware selection rather than vice versa. Therefore, in the allocation to the system, the cost of developing software makes it attractive to allocate certain functions to hardware, particularly those functions that are simple, independent of other software functions, and can be expected to remain the same over a long span of the system life cycle. This trend has caused the emergence of a new discipline — firmware engineering. The software allocations are made in situations whereby the control concept is not stable nor

well defined and whereby data handling tasks predominate. Functions are allocated to software where changes are anticipated over the life of the system. Also, and more important, we allocate to software those functions that integrate all the elements of a system into one smooth operation" (reference 15).

The results of this task are documented in a report which is used in the development of the hardware and software specifications.

1.24 Establish Initial System Operating/Control Procedures. The system operating and control procedures involve not only those procedures associated with the operation and use of the digital avionics and flight control systems themselves, but also the procedures associated with the operation of the vehicle and its subsystems in the airspace. This involves definition of the minimum equipment list required for dispatch, maintenance procedures in case of a detected fault, and the more complicated operating procedures necessary when operating in the constricted airspace and attempting to utilize time-of-arrival algorithms. For example, if the crew wishes to minimize cross-track deviation, they can command heading changes as a function of bearing. This results in trading off the smaller cross-track deviations for greater time along-track deviations. Various tradeoffs between system operating procedures are possible just as there are tradeoffs in the design. The result of this activity should be a report documenting the tradeoffs used and the recommended system operating and control procedures.

1.25 Review Reliability Analysis. This activity is conducted by the FAA and consists of a thorough review of the analysis provided by the manufacturer. The results of this review should be provided to the manufacturer so that he may correct any discrepancies or provide any additional information required.

1.26 Review System Safety Assessment Analysis. The system safety assessment analysis performed by the developer will be furnished to the FAA for their review. The FAA will submit any questions concerning the system safety assessment to the manufacturer upon completion of the review.

1.27 Prepare System Hardware Design/Interface Specification(s). This activity involves writing the appropriate system hardware design specifications and the interface specifications describing all hardware interfaces between subsystems. All of the reports previously prepared, as well as the system specification, should be available for this activity. The output is the individual hardware design specifications as well as the interface specifications.

1.28 Complete System Software Requirements Definition. The inputs to this task are the preceding reports which define all software functions, the fault tolerant algorithms recovery procedures, the switching logic functions allocated to software, and initial system operating and control procedures. The software architecture will be defined and the software functions to be performed by each processor defined in terms of their:

- (1) Control structure,
- (2) Data structure,
- (3) Data flow control, and
- (4) Application structures.

The control structure can be divided into the operating systems in the executive. The operating system will allow multiple processors to access global memory in designs using this architecture. The operating system functions of:

- (1) Request handling/interrupt control
- (2) Task control (scheduling and dispatching)
- (3) Resource allocation
- (4) Fault monitoring

should be described. The data base, data flow control in a distributed system, and the application modules which implement the system functions should be described. The application's functional descriptions should include the input, algorithms to be used, accuracy, constraints, and output.

The system software development specification will describe the overall system software requirements. This specification will be the primary reference document for all systems software. Software located in individual processors must be traceable back to the system software development specification.

1.29 Validate System Design Against Requirements. This activity is the system design review which evaluates all work leading up to this point in the system development. "The review is conducted when the definition effort has proceeded to the point that requirements and design approach have achieved a precise level of definition. This normally occurs at the end of the validation phase or early in the full-scale development phase of the system life cycle" (reference 16). The input to this design review includes the system specification, system hardware design specifications, system hardware interface specification, and system software development specification.

The output of this activity is a system design review report. At this point, the allocated configuration baseline is complete.

10.2.3 System Design Phase Activities.

2.1 Select/Design Avionics/Flight Control Hardware. The system design may make use of some existing or "off-the-shelf" components as well as require the complete design and development of new components and subsystems. This activity involves an analysis and experimentation to determine which existing components may be candidates and to select from these components. In addition, the design for new components must be completed. Mathematical models and simulation are often used in evaluating the performance and reliability characteristics of existing equipment as well as in the design of new equipment. The primary difference is the depth of detail required since the design of a new item entails working with piece part component and individual integrated circuit characteristics as well as the design of components, such as large-scale integrated circuits or hybrid circuits that incorporate the functions of many individual piece part components. The preferred system design is the one with the lowest cost that meets the specified minimum acceptable performance effectiveness value. Alternatively, if a maximum reliable system cost is specified, a preferred system is the one with the maximum performance effectiveness that does not exceed the cost limit. In the design of new subsystems, it is necessary to perform the electric packaging design including mechanical, thermal, and other environment modifying design techniques such as sealing and pressurizing the line replaceable unit. To complete the hardware design process, the specification and drawings must be updated to reflect the specific characteristics of each component of the selected system configuration.

The output of this activity is the design specification and documents for each subsystem.

2.2 Design System Integration/Support Facility. The system's integration/support facility is a tool to be used for development and integration of the avionics and flight control systems. This activity involves establishing the requirements, characteristics of the facility, development of a program plan for the development of this facility, followed by the complete development of the facility. The facility processors shall host the support software required for development, test, and integration of the object code for each processor used in the avionics and flight control system. The support facility host processors will host the compilers, assemblers, linkers, editors, and loaders for the flight processors. Many support facilities include one or more processors capable of emulating the microcode of the actual flight processors (reference 17).

In addition to the processors, the support facility shall include the network interconnecting the support facility processors, the data collection instrumentation, and other associated electronic test instrumentation. The support facility generally contains a test control center for controlling the use of the simulators in the support facility. Many manufacturers have developed support facilities (references 18 and 19). Documentation on these facilities' system architecture, system hardware, and system software should be available to the regulatory agency. The documentation should include, as a minimum, the program plan and system subsystem level specifications such as available for the USAF Flight Engineering Facility (references 20, 21, and 22).

2.3 Prepare Software Development Plan. The software development plan describes the:

- (1) Software development tasks and schedule including milestones, reviews, key meetings, audits, documentation releases, and product deliveries.
- (2) Methodology — a summary of the practices, standards, and techniques to be employed in developing the product. These are essentially technical items such as programming language and allowable logical control structure.
- (3) Configuration item and deliverables.
- (4) Configuration management plan.
- (5) Applicable documents and documentation standards.

The software development plan should make allowances for correction of errors found through testing and retesting to verify that the errors have been corrected and no new errors introduced. This plan shall be the primary management document for the subsequent software development.

2.4 Define Computer Program Configuration Items Requirements. The objective of this set of activities is to develop detailed Computer Program Configuration Item (CPCI) specifications. These specifications are a statement of the development requirements for each CPCI, whether they are routines, programs, groups of programs, or the entire software subsystem (if it is small).

The individual CPCI specifications shall be traceable to the software development plan, configuration item index, and the system software development specification.

2.5 Develop System Software Interface Specification. The system software interface specification describes in detail the requirements for all data transmitted between digital subsystems. The format of each word and, in multiple word messages, the format of each message is totally specified. If the data transmissions are currently on a synchronous basis, the transmission rate is specified. If a command response protocol is used in which addresses and subaddresses are used for communication rather than a broadcast protocol, the addresses and subaddresses of each message or word is given. This specification serves as a basic software interface control document and should be under configuration control. Any data transmissions between subsystems other than those prescribed in the software interface specification should be invalid.

The system software interface specification should be provided to the regulatory agency as well as to the customers of the manufacturer.

2.6 Develop/Modify System Integration/Support Facility. If no system's integration/support facility exists, the manufacturer must develop a facility which meets requirements previously formulated in the design of the system's integration/support facility. Should the manufacturer presently have a system integration/support facility, the activity may merely involve making minor hardware modifications or modifying software data acquisition programs to acquire the test data to be obtained during the system test and integration. At the conclusion of this activity, the system's integration support facility should be complete including all hardware and software.

The output of this activity is updated documentation reflecting the actual configuration and capabilities of the system's integration/support facility.

2.7 Define Test Requirements. The objective of this effort is to define and document the test requirements. The system should have been designed from the beginning to be testable. The test requirements document describes the software test approach and addresses:

- (1) The testing philosophy followed,
- (2) Responsibility for the various levels of testing,
- (3) Software performance measures and standards,
- (4) Method to be followed in handling software change proposals emanating from testing activity, and
- (5) Test report requirements.

The test requirements document is used for the detailed test planning and development of test procedures for each test plan.

2.8 Commence Software Real-Time Operating System, Programs, and Modules Design. The objective of this activity is to develop the design specification for each individual software program, module, or routine. The individual software

design documents are the basis for the preliminary design review for that component of the software. Using a structured design procedure (references 23 and 24), each module is designed using the allowed basic constructs and the algorithm defined in the CPCI development specifications.

CPCI software design documents must identify each module, the module's data flow, associated structure diagram, and the associated data tables.

2.9 Hardware Preliminary Design Review. The inputs to the hardware preliminary design review are the individual hardware design description documents and the hardware development specifications. The purpose of the design review is to review areas such as hardware trade-offs and functional interfaces, errors due to lack of understanding of the critical design areas, and the interfaces of the system's integration/support facility with each of the hardware items. The results of the preliminary hardware design review are discrepancy reports which document the agreed-to corrective action.

2.10 Verify Software Requirements Against System Requirements (Software Preliminary Design Review). The Preliminary Design Review (PDR) "is a formal technical review of the basic design approach for a CPCI. There would normally be one successful PDR for each CPCI. A collective PDR for a functionally related group of CPCIs, treating each CPCI individually, may be held when such an approach is advantageous. The PDR is held after authentication of the CPCI Development Specification and the accomplishment of preliminary design efforts, but prior to the start of the detailed design" (reference 25).

The software development support tools planned for use during program development should be reviewed for completeness. The test requirements document should be reviewed along with the CPCI Development Specification to assure agreement.

The responsibility for conducting the design review rests with the individual responsible for the design activity. "During the review, the reviewers are expected to comment, first, on the completeness, accuracy, and general quality of the work product. Major concerns are expressed and identified as areas for potential follow-up. The designers present a brief overview of the product. They then 'walk' the reviewers through the design in a step-by-step fashion that simulates the function under investigation. They attempt to review the material in enough detail so that the concern expressed at the beginning are either explained away or identified as action items. Significant factors that require further action are recorded as they are identified" (reference 26).

After resolution of the action items, the resultant design is released into the control cycle, according to prescribed configuration control methods (reference 26).

2.11 Review Specification Document. This activity is conducted by the regulatory agency and consists of a complete review of the system specification, system software development specification, computer program configuration item development specifications, system software interface specifications, system hardware design specification, and system hardware interface specification. The regulatory agency will review the documentation for completeness and consistency between specifications. Any discrepancies noted will be documented and provided to the developer.

2.12 Update Specification and Drawings. After completion of the hardware preliminary design review, the manufacturer should update all specifications and drawings to reflect any changes resulting from the design review action items. The updated specifications and drawings are then used in the subsequent final design phase.

2.13 Breadboard and Evaluate Each Circuit. The manufacturer performs the final design of the hardware comprising the subsystems. This is likely to require breadboarding and evaluation of any new circuits. In addition to the performance evaluation, the manufacturer may acquire samples of the components planned for use in the production equipment. These samples would be subjected to the component test previously described in Section 9.

The output of this activity would consist of recommended changes to the baseline established at the hardware preliminary design review, documented as updates to specifications and drawings.

2.14 Hardware Critical Design Review. The inputs to the hardware critical design review are the recommended updates to the preliminary design baseline based upon the design evaluation and performance tests. These recommended changes are considered by the reviewers and either approved or noted as an action item requiring resolution. Once these action items are resolved, specifications are updated to reflect the design baseline which will be used by configuration management in the subsequent phases.

2.15 Develop System Validation Test Plan and Procedures. The system validation test plan and procedures should be developed by the manufacturer after consultation with the regulatory agency. As previously stated, validation encompasses verification. The validation test plan should describe the techniques or methods to be used in the validation of the system. Sections 3, 8, and 9 of the handbook describe various methods including those associated with design validation, hardware testing, software testing, and system level tests. The validation test plan should specifically identify each of the selected test concepts which will be used for the foregoing.

The validation test plan will contain the test objectives, and test description, description of the test environment, including required hardware and software, the delineation of the requirements being validated, and an evaluation plan. The evaluation plan will consist of the acceptance criteria and a description of the techniques to be used in analyzing the test data in order to determine compliance with the acceptance criteria.

Individual test procedures will describe the sequence for specific tests, the test input data, the data base, identify the software configuration, and identify the required test personnel and their functions.

Observations of the test itself and evaluation of the test output data constitute the basis on which it is determined whether the test objectives have been met, pertinent requirements validated, and the acceptance criteria satisfied. The evaluation of the output data, if performed manually, is likely to be a tedious and time-consuming process for all but the most elementary of tests. The manual task of error-checking is in itself an error-prone process (reference 27).

The system validation test plan and procedures are used by the manufacturer and the regulatory agency during the validation and certification of the system.

2.16 Develop Test Plans. This activity involves developing the test plans for each of the test levels including:

- (1) Stand-alone testing of modules,
- (2) Software integration,
- (3) System integration, and
- (4) Flight test.

Each test plan specifies the methodology to be employed. As stated in Section 8, the test plan traces the testing sequence from unit level test to final acceptance test and identifies each individual test. Test procedures, keyed to the test plan provide step-by-step instructions for the execution of the test and specify precisely what outputs are to be expected.

Test support software or the hardware test bed to be used should be identified as well as all testing inputs (reference 28, 29, and 30).

Detailed test procedures as described in previous activities should be developed for each test plan. The test procedures shall be sufficiently detailed that they can be used in the complete integration, replication, and validation of this system software. Test procedures must also provide all information required for integration of the system and flight test of the system.

The test plans and procedures shall be furnished to the regulatory agency for their review prior to the critical design review.

2.17 Perform Detail Software Design. As stated in Section 8, final design is often done using a formal design methodology such as the structured design (references 23 and 24) or other methods. During the final design effort, a design walk-through should be used by the developers to verify the flow and logical structure of this system while design inspections should be performed by the test team.

The output of the final design phase is the detailed design document which is the basis for the critical design review.

2.18 Verify Software Design against Software Requirements (Critical Design Review). "The critical design review (CDR) is a formal technical review of the CPCI detailed design conducted prior to the start of coding. The CDR is intended to ensure that the detailed design solutions, as reflected in the draft of CPCI products specification, satisfy performance requirements established by the CPCI Development Specification. The CDR is also accomplished for the purpose of establishing integrity of computer program design at the level of flow charts or computer program logical design prior to coding and testing. The principal items reviewed are the complete draft of the CPCI Product Specification and drafts of test plans/procedures. All changes to the CPCI Development Specification and available test documentation are examined to determine compatibility with the test requirements of the Development Specification" (reference 25).

After resolution of any action items resulting from the design review, the resultant design is released to configuration control and becomes a software design baseline.

10.2.4 System Full-Scale Development Activities.

3.1 Build/Test Prototype Hardware Subsystems. The tests in this activity related to the validation are primarily those related to component screening and acceptance testing and the environmental qualification testing. In addition, failure modes and effects tests should be conducted at the individual subsystem level to verify the system redundancy designed for the classes of faults the system is to tolerate. The individual test reports should be submitted to the regulatory agency for its review. Any discrepancies identified in the tests should be analyzed and modifications required to make the system operate properly identified and submitted to the change control board.

3.2 Integrate/Test Hardware System's Subsystems. As discussed in Section 9, a sequence of integration tests should be performed to integrate each of the hardware subsystems. A simulator may be used in this testing to provide the test drivers signals for items not yet integrated.

The output of each integration step in the sequence is a test report which documents any discrepancies or anomalies noted as well as those test procedures which were successfully completed.

3.3 Code Modules and Debug. This activity involves the actual coding in the selected language and debugging of the code. As discussed in Section 8, code walk-through and code inspection are manual techniques. Assembling or compiling the code also provides a debug for those errors the compiler or assembler is designed to detect. Errors found during the debug should be corrected before beginning coding of another module.

3.4 Perform Stand-Alone Module Testing. As discussed in Section 3 and 9, the stand-alone test may use the techniques of:

- (1) Static analysis,
- (2) Dynamic testing without or without instrumentation probes,
- (3) Symbolic execution, and
- (4) Proofs of correctness.

Code execution testing may be done on a host computer which simulates or emulates the target computer or the actual execution may be done on the target machine as discussed in Sections 3 and 8.

Whichever module testing approach is taken, one basic criteria for the set of test cases is to ensure that they cause every instruction in the module to be executed at least once. All logical paths should also be traversed. The testing should be done in the sequence specified by the test plan and procedures. The result of the stand-alone test should be documented in a stand-alone test report noting any discrepancies which will necessitate retesting.

3.5 Verify Code Versus Design. This activity is conducted by the independent test organizations. A walk-through or inspection may be used as described in Section 8 of the handbook. In addition, a static analyzer may be used by the independent test organization. The independent test organization shall document the results of verification of the code versus the design in a report noting any discrepancies. This report will be furnished to the software developer.

3.6 Test Modules. This test is done by the independent test organization and involves an independent test of the modules previously tested by the manufacturer. The independent test organization is likely to use a dynamic analyzer and execute the code for each of the modules. The data collected by the instrumentation probes in the dynamic test mode will be analyzed and a test report prepared noting any anomalies. This test report will be furnished to the developer of the software.

3.7 Integrate Modules and Test Each CPCI. The manufacturer shall integrate the modules using one of the methods described in Sections 3 and 8. The test plan should have specified which of the seven approaches (bottom-up testing, top-down testing, modified top-down testing, big-bang testing, sandwich testing, modified sandwich testing, and thread testing) is to be used.

Integration testing is primarily functional with the main emphasis on the interaction between the software components and their interfaces. The testing often takes place in the laboratory containing the target computers and enough equipment to simulate the application with considerable fidelity. As each test is conducted, a test report will be generated. After all testing is completed, the final report is generally prepared which includes all errors detected and the status of their correction.

3.8 Test Module Integration/CPCI. This activity is conducted by the independent test organization with the purpose of verifying interfaces, computational accuracies, timing, and sizing. While some of the tests may be run using an emulation of the target processor and the instrumentation probes, final module integration test for each CPCI should be tested in the actual computer and hardware environment. These tests will be run under "live" conditions using test drivers in the avionics integration support facility.

Any discrepancies or anomalies noted during the independent module integration tests will be documented and furnished to the software developer.

10.2.5 System Integration/Test Activities.

4.1 Integrate System (Software and Hardware) and Test in Laboratory. The system developer will integrate and test the system in accordance with the system test plan. This will be done in the laboratory and make use of simulation facilities as the system is sequentially integrated as discussed in Sections 3, 8, and 9. The results of each integration step and test should be documented in a system integration report.

4.2 Test System Integration. This activity is conducted by an independent test organization in accordance with the system's integration/test plan. Failure modes and effects tests are often conducted in each integration step by the independent test organization. Extensive use is made of simulation facilities as discussed in Sections 3, 8, and 9 of this handbook.

The independent test organization shall prepare a report documenting any discrepancies for each integration step and an overall report summarizing all discrepancies noted.

4.3 Integrate System into Aircraft. This activity is conducted by the manufacturer as discussed in Sections 3, 8, and 9. In some cases, the test aircraft or

avionics systems may not necessarily be the only aircraft the system will ultimately be used in. In this case, the aircraft interface must be specifically noted and special instrumentation may be required if it is expected that the interface in another aircraft could be greatly different. At the completion of this test, a product baseline will have been fully defined, which then becomes the baseline used by the configuration management organization.

The output of this activity is an aircraft integration test report noting any problems or discrepancies observed in the integration of the system into the aircraft.

4.4 Perform System Validation Test. This test will be performed by the independent test organization and, as stated in reference 31, involves:

"(1) Demonstrating compliance with system requirements under operational conditions and range utilizing simulated aircraft characteristics, and

(2) Demonstrating system compliance with specific environmental conditions."

The system validation tests are designed to demonstrate that the system will correctly operate in the environment it is designed to operate in and tolerate system transients and other faults the system was designed to tolerate. These independent validation tests may occur in the same time frame as the flight tests performed by the manufacturer.

Any discrepancies or anomalies identified during the validation will be documented and provided to the manufacturer.

4.5 Perform Flight Tests. The manufacturer will perform flight tests in accordance with the flight test plan and under the cognizance of the regulatory agency. Should the flight tests reveal the need for change in the hardware or software, the change would normally be made and validated in the avionics integration support facility as previously done before flight testing. At the completion of the flight test, a functional configuration audit may be performed on the software. The functional configuration audit (FCA) "verifies that the CPCI's actual performance complies with the requirements of the development specification. Data from tests of the CPCI is perused to verify that the item has performed as required. FCAs may be conducted on an incremental basis. The FCA for a complex CPCI may be conducted on a progressive basis with completion of the FCA occurring after the completion of the level of integrated testing that may be required to validate the CPCI. Requirements of the Development Specification not validated by the CPCI test are identified, and a solution for subsequent validation is proffered (such as validation in the subsystem for system test). An audit of the test plans/procedures are made and compared against the official test data, including checks for completeness and accuracy. Deficiencies are documented, and completion dates for all discrepancies are established and recorded. An audit of the test report is performed to validate that data accurately and completely describe the CPCI test" (reference 33).

After completion of the engineering flight tests, identified discrepancies are corrected and retested. The results of the flight test are documented and made available to the regulatory agency.

4.6 Perform Acceptance Tests. Acceptance tests are usually conducted in one of two ways. In the first case, the test is conducted by the developing organization while being witnessed by the customer or user of the system. The customer in this case witnesses the execution of the actual test and verifies that the test results conform to those expected. If this system meets all of the customer requirements, the customer normally signs off and accepts the system.

In the second case, the test may be performed by the program's customer or end user. In this case, "The best way to do this is to devise test cases attempting to show that the program does not meet the contract; if these test cases are unsuccessful, the program is accepted. In the case of a program product (e.g., the computer manufacturer's operating system or compiler, or a software company's data base system), the sensible customer first performs an acceptance test to determine whether the product satisfies its needs" (reference 32).

At the completion of the acceptance testing, the Physical Configuration Audit (PCA) is conducted. At this point, the customer has the final draft of the CPCI Product Specification. "The PCA includes an audit of the Product Specification and an inspection of the format and completeness of the user's manual and any other manual or handbooks due for acceptance at this time. Other specific review tasks also include: examine Product Specification for format and completeness and scrutinize the PCA minutes for discrepancies that require action" (reference 33).

At this point, an acceptance test report should be written noting any discrepancies identified during the acceptance test or the Physical Configuration Audit.

4.7 System Certification. The system certification will be conducted in accordance with the certification test plan agreed to by the manufacturer and the regulatory agency. This activity is a culmination of the proceeding activities. The final certification tests are flight tests. Prior to or during the flight test, actual data collected shall be compared with that used in early analyses such as the reliability and safety assessment analyses and determine if there is a great discrepancy in the data and a need to redo the analyses. The results of each of the tests conducted ranging from the individual software module tests up through the software integration and system integration tests performed using the simulation support facilities shall be reviewed. As stated in Section 8 in reference 34, results of simulation tests are being used as a substitute for the many costly hours of flight tests where the simulation can be shown to yield valid results. In many cases, the simulators are used for conducting hazardous or high risk tests instead of actual flight tests (reference 34).

Reference 34 investigated the feasibility of using simulation in place of flight tests for testing and certification of the navigation, navigation support, warning, autopilot performance and malfunction, autothrottle, autoland and go-around, and flight director. This study concluded that, depending upon the amount of data available in the data base, use of simulation of certification was technically feasible for:

- (1) 25.1301 Function and Installation
- (2) 25.1303 Flight and Navigation Instruments
- (3) 25.1309 Equipment Systems Installation
- (4) 25.1321 Arrangement and Visibility
- (5) 25.1323 Airspeed Indicating System
- (6) 25.1325 Static Pressure System

- (7) 25.1327 Magnetic Direction Indicator
- (8) 25.1329 Automatic Pilot System
- (9) 25.1331 Instruments Using Power Supply
- (10) 25.1431 Electronic Equipment
- (11) 25.1457 Cockpit Voice Recorders
- (12) 25.1459 Flight Recorders.

At the completion of the system certification tests, the regulatory agency will issue the appropriate type certification if the tests were satisfactory and completed.

4.8 Perform Operational Test and Evaluation. The Operational Test and Evaluation (OT&E) will be conducted in accordance with the test plan. The objectives of the OT&E tests are to determine the operational effectiveness and operational suitability of the system. The operational effectiveness portions of the test are concerned with the capability of the system to perform its intended function in the operational environment while the operational suitability is concerned with the degree the system supports a mission and is maintainable (reference 35). These tests are normally conducted by the end user. The results of these tests are used for identification of required modifications to the system hardware or software. These results are furnished to the manufacturer in the form of reports for use in correcting the discrepancies noted.

10.2.6 Production and Deployment Activities.

5.1 Production and Deployment. These activities consist of production of the quantities of the system required by the user, the acceptance testing of each system by the user, and the introduction into operation of each of the new systems as they are delivered from the manufacturer.

10.2.7 Operation and Maintenance Activities.

6.1 Operation and Maintenance. The user of the system must continue the configuration management activities. "Changes to system functional capability required by the user where the discovery of design errors during service will necessitate post-certification software changes. Such changes can lead to 'secondary errors' in the software; i.e., errors that were not present, or whose affects were not detected, when the system was first certificated. Thus, careful consideration must be given to verification/validation of the changes. The goal of this effort should be to ensure the same level of confidence in the software after the change as achieved during the original precertification test program" (reference 36).

Similar considerations apply to hardware subsystem modification.

It is recommended that users of this handbook establish a formal data code collection data base system for the digital systems. This information will be of great use in the maintenance of the digital systems hardware and software.

6.2 Recertification Criteria. The current regulatory guidelines for determination of when a digital system requires recertification are undergoing review. Additional criteria related to the impact of changes necessitated by additions to the system, hardware and software will be useful to the manufacturer and the user. The recertification criteria will probably be announced in the Federal Register in the form of an advisory circular in a manner similar to that for AC 25.1309-1.

6.3 System Revisions and Implementation of Engineering Change Notice (ECN)/ FAA Advisory Directive (AD). The user must make revisions to the system as required by advisory directives. In addition, if the user plans to change a system configuration, an Engineering Change Notice (ECN) will be prepared for use in implementation of the configuration change. This applies to both hardware changes and software changes. Those changes requiring recertification will be submitted to the regulatory agency in the form of documentation including test plans and test results.

6.4 User Modifications. The user may, through his maintenance center, make the needed modifications to nonessential or essential functions' hardware and software. It is possible that the user may be certified to make modifications to critical functions. It is also possible that these modifications would be made for the user by the manufacturer. In either case, the modifications to both the hardware and software must be fully documented in accordance with the configuration control procedures.

If recertification is not required, the modified system is put into operation. If recertification is required, the system functions criticality category must be agreed on. This may involve the manufacturer and the regulatory agency as shown in figure 10-4. The recertification may involve repeating many of the steps performed in the original system certification if the changes are to flight critical functions.

10.3 REFERENCES.

1. Hill, J. D., and Warfield, J., A Unified Systems Engineering Concept, Battelle Memorial Institute, Columbus, Ohio, June 30, 1972.
2. Dickman, S., and Katzenbarger, F. X., Systems Engineering, Sperry Rand Engineering Review, Vol. 21, No. 1, 1968.
3. Willis, R. R., DAS An Automated System to Support Design Analysis, Pages 109-115, Proceedings of 3rd International Conference on Software Engineering, May 10-12, 1978, IEEE Catalog No. 78CH1317-7C.
4. Stephens, S. A., and Tripp, L. L., Requirements Expression and Verification Aid, Ibid, Pages 101-104.
5. Schindler, M., Today's Software Tools Point to Tomorrow's Tool Systems, Electronic Design, Vol. 29, No. 15, July 23, 1981, Pages 73-110.
6. Airplane System Design Analysis, Advisory Circular 25.1309-XX, Department of Transportation, Federal Aviation Administration, Federal Register, Vol. 46, Issue 214, November 5, 1981, Pages 54958.
7. Stiffler, J. J., et al, CARE III, Final Report, Phase I, NASA CR 159122, November 1979.
8. Smoliar, S. W., Operational Requirements Accommodation in Distributed System Design, IEEE Transactions on Software Engineering, Vol. SE-7, No. 6, November 1981, Pages 531-557.
9. Dhillon, B. S., and Singh, C., Engineering Reliability, New Techniques and Applications, Wiley-Interscience Publication, John Wiley & Sons, New York, 1981, Pages 43-45.
10. Software Considerations in Airborne Systems and Equipment Certification, Document No. RTCA/DO-178, Radio Technical Commission for Aeronautics, November 1981.
11. Hamilton, M., and Zeldin S., Higher Order Software--A Methodology for Defining Software, IEEE Transactions on Software Engineering, March 1976.
12. Kobayashi, H., Modeling and Analysis, An Introduction to System Performance Evaluation Methodology, Addison-Wesley Publishing Company, Reading, Massachusetts, 1978, Page 221.
13. Jensen, R. W., and Tonies, C. C., Software Engineering, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1979, Pages 195-196.
14. Kobayashi, H., Modeling and Analysis, An Introduction to System Performance Evaluation Methodology, Addison-Wesley Publishing Company, Reading, Massachusetts, 1978, Page 221.

15. Jensen R. W., and Tonies, C. C., Software Engineering, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1979, Pages 125-132.
16. Jensen, R. W., and Tonies, C. C., Software Engineering Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1979, Page 398.
17. Clark, N. B., The Total System Design Methodology, Rome Air Development Center, Griffis Air Force Base, New York.
18. Spradlin, R. E., The 757/767 Flight Management System Laboratory Test Program, 4th AIAA/IEEE Digital Avionics Systems Conference, November 1981.
19. Uring, H. R., DC-9 Super 80 Digital Flight Guidance System Simulation Techniques for Certification, Proceedings of the 1981 RTCA Assembly, Radio Technical Commission for Aeronautics, Pages 57-69.
20. Hitt, E. F., AFFDL/DAIS Flight Engineering Facility Program Plan, PF000 000, TRW Defense and Space Systems Group, April 15, 1977.
21. Hitt, E. F., AFFDL/DAIS Flight Engineering Facility System Specification, SF 000100, TRW Defense and Space Systems Group, April 1977.
22. Arnold, J., Prime Item Development Specification for Digital Aircraft Simulator, SF 000200, TRW Defense and Space Systems Group, October 18, 1977.
23. Stevens, W. P., Myers, G. J., and Constantine, L. L., Structured Design, Pages 114-122.
24. Dolbey, S. C., and Cary, D. R., Management of Software Design--A Structured Approach, Proceedings of the IEEE 1981 National Aerospace and Electronics Conference, Pages 172-178.
25. Jensen, R. W., and Tonies, C. C., Software Engineering, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1979, Page 399.
26. Jensen, R. W., and Tonies, C. C., Software Engineering, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1979, Page 216.
27. Jensen, R. W., and Tonies, C. C., Software Engineering, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1979, Pages 341-342.
28. Hartwick, R. D., Test Planning, Logicon, Incorporated.
29. Panzl, D. J., Test Procedures: A New Approach to Software Verification, Proceedings Second International Conference on Software Engineering, October 1976, Pages 477-485.
30. Rubey, R. J., New Approaches for Software Validation, NAECON '72 Record, Pages 252-258.

31. Software Considerations in Airborne Systems and Equipment Certification, Document No. RTCA/DO-178, Radio Technical Commission for Aeronautics, November 1981, Pages 35-42.
32. Myers, G. J., The Art of Software Testing, John Wiley and Sons, New York, 1979, Page 119.
33. Jensen, R. W., and Tonies, C. C., Software Engineering, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1979, Page 400.
34. Archibald, D. M., The Role of Simulation Methods in the Aircraft Process, Report No. FAA-RD-77-17, Lockheed-California Company, March 1977.
35. Murch, W. G., Operational Test and Evaluation of Software, Proceedings of the IEEE 1981 National Aerospace and Electronics Conference, Pages 1390-1398.
36. Software Considerations in Airborne Systems and Equipment Certification, Document No. RTCA/DO-178, Radio Technical Commission for Aeronautics, November 1981, Page 34.

SECTION ELEVEN

RECOMMENDED CONFIGURATION MANAGEMENT PROCEDURES

SECTION ELEVEN

RECOMMENDED CONFIGURATION MANAGEMENT PROCEDURES

TABLE OF CONTENTS

	Page
SECTION 11	11-1
11. RECOMMENDED CONFIGURATION MANAGEMENT PROCEDURES	11-1
11.1 Configuration Management Plan	11-1
11.1.1 Configuration Identification	11-3
11.1.2 Configuration Control Board	11-4
11.1.3 Configuration Status Accounting	11-6
11.2 Software Configuration Management	11-7
11.3 Summary	11-8
11.4 References	11-9

SECTION 11

11. RECOMMENDED CONFIGURATION MANAGEMENT PROCEDURES

The discussion in this section covers both the software and hardware elements of the system. While all hardware configuration management practices might be thought totally applicable to software, a difference exists in that it is much easier to modify software than it is hardware. Therefore, some additional management procedures may be required for software as contrasted to hardware.

This section discusses the configuration management plan, the configuration identification, configuration tracking, and the configuration change control board.

This section concludes with the discussion of software configuration management procedures.

11.1 Configuration Management Plan.

There should be a configuration management plan for the system, the hardware, and the software. "The system configuration management plan is written at the system level and deals with, in a general way, the topics to be addressed, and their relative importance, within the hardware and software configuration management plans. Naturally, in a strictly software (or hardware) project, one configuration management plan will normally suffice. Figure 11-1 shows the progression of configuration management planning documents as it relates to the system life cycle. Note that although the system configuration management plan should be prepared immediately after project initiation, certainly well before the formal establishment of any baseline, the respective hardware and software configuration management plan should be prepared only after the allocated baseline has been approved. This approach will ensure the appropriate level of configuration management involvement in both hardware and software implementation activities." (reference 1)

As shown in Section 10, a system configuration management plan must be prepared at the beginning of the system life cycle. Before any baselines are approved, subsidiary hardware and software configuration management plans should be developed after approval of the allocated baseline.

"The first major topic which should be addressed in the system configuration management plan is that of the configuration control board (CCB). The plan should definitively outline the interaction and involvement of the various buyer, seller, and integrated CCBs. It is imperative that both buyers and sellers understand what they should expect from their CCBs very early in the development process. The specific role of the CCB in 'evolutionary' change control should be dealt with in the CM plan.

"The second major configuration management plan topic should be a complete description of the planned CM tools to be used on the project at hand. Under tools we include the forms used for change control, trouble reporting, and so on, conventions for labeling configuration items, as well as automated aid available to support the CM process. The definition of these tools will vary in detail between

the system CM plan and the hardware and software CM plan. Thus, an automated software library tool with particular applicability to software CM need not be defined in detail in the system CM plan.

"The third major CM plan topic deals with procedures. This topic is the heart of the CM plan since it addresses not only how the CM organization will deal with the rest of the project team but also how it will operate internally. For example, in just what order are system configuration items (SCI) identified, controlled, and audited? How does the CCB schedule an audit? How does status accounting provide information to the rest of the project team?

"The fourth, and final, major configuration management plan topic deals with resources and their allocation. Each CM plan should contain a budget that allocates resources by CM disciplines and by the life cycle stage" (reference 1).

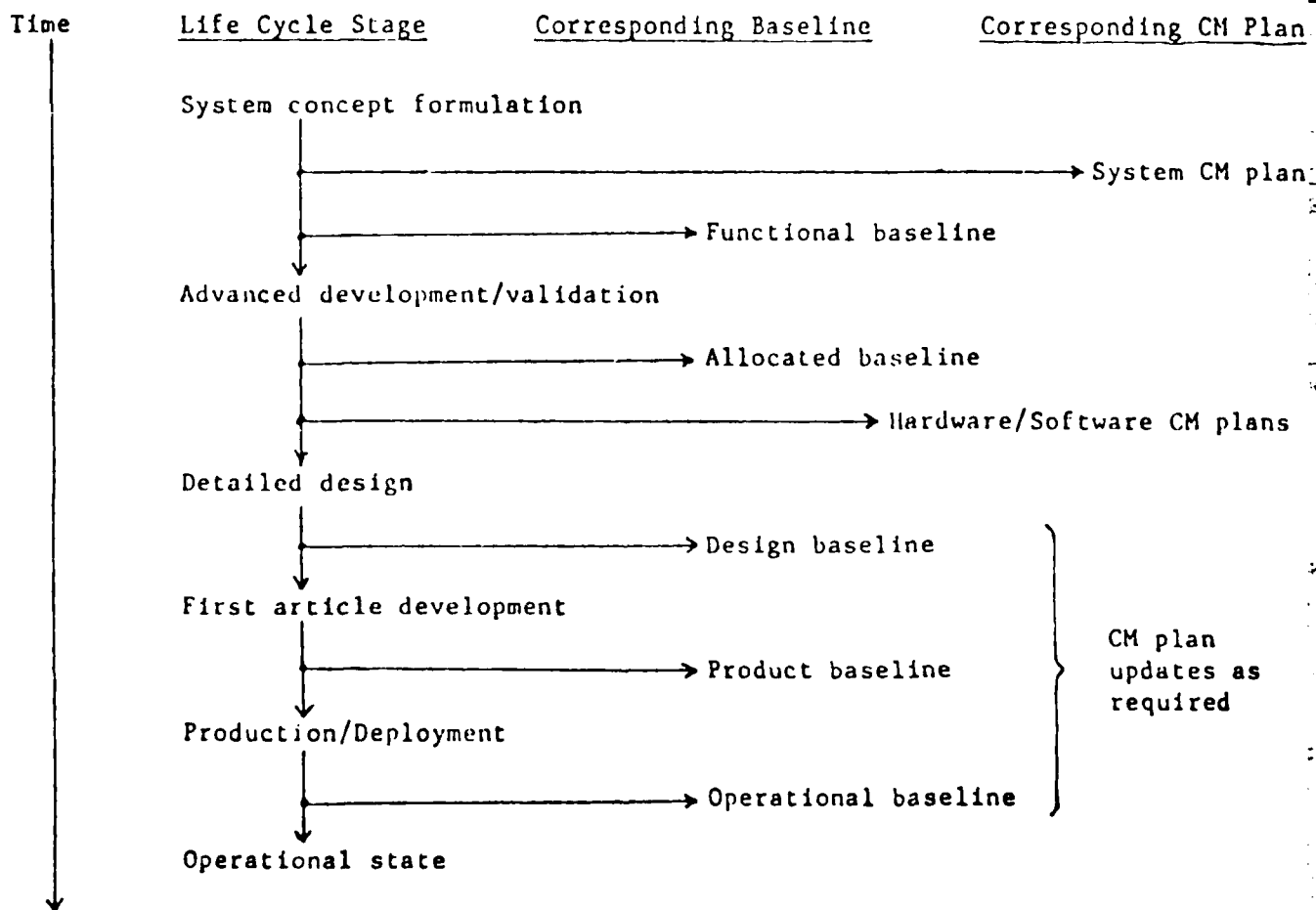


FIGURE 11-1. CM PLANNING WITH RESPECT TO THE LIFE CYCLE (Reference 1)

Figure 11-2 presents the basic outline of the configuration management plan. Note that the organization does not specifically identify the baselines. The baselines previously presented in Section 10 are felt to be a reasonable set of baselines. The more baselines there are, the more often the Configuration Control Board must meet to review changes to these baselines. As stated in reference 1,

"have as few baselines as reasonable for given projects and issue formal updates only when necessary (i.e., when the number of changes has become difficult to control)."

SCM PLAN

1. System Description
 - An overview of the system being built
2. CM Organization
 - The CCB (all of them)
 - Identification
 - Control
 - Auditing
 - Status Accounting
 - Other product assurance disciplines
3. CM Tools
 - Identification tools and labeling conventions
 - Control tools
 - Auditing tools
 - Status Accounting tools
4. CM Procedures
 - Design stages
 - Development stage
 - Deployment stage
 - Operational/Maintenance stages
5. CM Resources
 - Budget (dollar)
 - Budget (staffing)
 - Budget (other)

FIGURE 11-2. CM PLAN OUTLINE

11.1.1 Configuration Identification.

"The basic purpose of configuration identification is to facilitate partitioning of baselines into progressively larger numbers of SCIs as the life cycle is traversed." (Reference 2)

Reference 2 suggests that a labeling scheme for SCIs be adopted in which the indices that label a particular SCI explicitly show its parentage. The explicit identification of parentage facilitates configuration verification (which ensures that what is intended for each SCI is specified in one baseline or update and is actually achieved in the succeeding baseline or update) and configuration validation (which ensures that the configuration of SCIs satisfies buyer/user requirements) (reference 3). Figure 11-3 illustrates how an SCI tree can grow and be fragmented across multiple documents. "A single document (e.g., the overall system concept) generates two or more documents which detail the system concept in specific areas (e.g., hardware design, software design, and data base design) and this collection of documents may, in turn, generate another collection of documents which detail these specific areas in greater depth (e.g., hardware logic diagrams, software logic diagrams, and data element specifications). Thus, the growth of the SCI tree is characterized by corresponding growth of a document tree, with individual elements of the document tree embodying parts of the SCI tree. Figure 11-3 also indicates how the labeling mechanism (S) CI_{x, y, z} ... explicitly links all elements of the document tree (hardware as well as software) (reference 4). The labeling mechanism should be relatable to the label used to identify predecessor products. The particular labeling mechanism chosen is not important as long as it satisfies the foregoing principle and is clearly described in the configuration management plan.

The labeling mechanism chosen applies not only to all documentation but also to the hardware and the actual software items. These labels should actually be stored as part of the code in memory as well as on any mass storage media as discussed later.

11.1.2 Configuration Control Board.

The Configuration Control Board is a formal board made up of personnel representing the manufacturer or developer, the buyer, and using organization. The Configuration Control Board approves, monitors, and controls conversion of design objects into system configuration items. The board also approves, monitors, and controls changes to the system. Seldom does this Configuration Control Board initiate a change. "More typically, changes develop as a result of new requirements, modifications in design, or, ultimately system deficiencies. Notifications of such situations reach the configuration manager in one form or another as requests for change. After a change request is formalized, it is sent to the appropriate body for analysis. It is examined, particularly by the developers, for potential schedule and budget impact, as well as for technical impact on other elements of the system. The results of this analysis are documented as an Engineering Change Proposal (ECP). The ECP is forwarded to the CCB by the configuration manager. It will evaluate the ECP on its merits, including urgency and impact and either approve it or not.

Approved ECPs are treated, by the developers, as changes in requirements and must be reflected in already existing baselines as changes or updates. The status accounting function of configuration management will provide feedback to those interested in the status of all ECPs whether approved or not" (reference 5).

Figure 11-4 depicts a typical change request form. Each manufacturer or developer may have his own form but the information contained should be of the type shown in the figure.

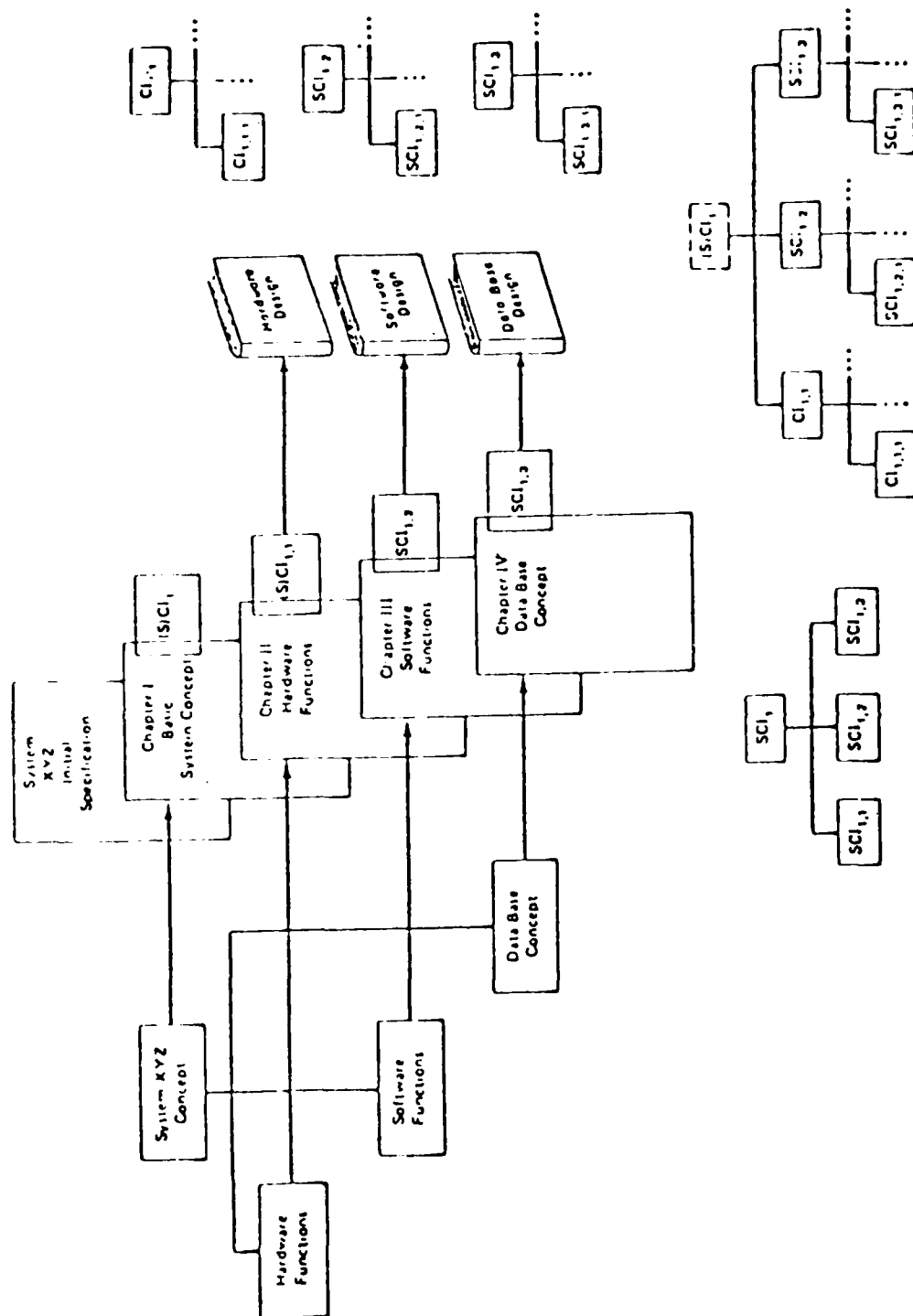


FIGURE 11-3. FRAGMENTING SCI TREE ACROSS MULTIPLE DOCUMENTS (Reference 4)

CHANGE REQUEST

1. System name: _____		2. Control no.: _____	
3. Application level: System <input type="checkbox"/> Hardware <input type="checkbox"/> Software <input type="checkbox"/> Document <input type="checkbox"/> Other <input type="checkbox"/>			
4. A. Originating organization	5. CI affected (highest level)	6. Documents affected A. _____ D. _____ B. _____ E. _____ C. _____ F. _____	
B. Initiator			
C. Telephone #	7. Priority: A. Routine <input type="checkbox"/> B. Urgent <input type="checkbox"/> C. Emergency <input type="checkbox"/>	8. Other systems/ software/equipment affected Yes <input type="checkbox"/> No <input type="checkbox"/> If yes, explain in block 9.C	
D. Date			
9. Narrative A. Description of change B. Need for change C. Estimated effects on other systems/software/equipment D. Alternatives			
To be completed by cognizant CM manager			
10. Date received	12. Disposition		
11. ECP requested Yes <input type="checkbox"/> No <input type="checkbox"/>	13. Signature	14. Date	

FIGURE 11-4. CHANGE REQUEST (Reference 1)

11.1.3 Configuration Status Accounting.

Configuration status accounting tracks and documents both requested changes as well as system problems and coordinates initiation of change with the execution of change. The principal function of configuration status accounting are data recording, data storing, and data reporting. The data recording function basically consists of keeping a record of what happened and when. The data recording

function involves manual or automatic recording of data that supplies the information from which the configuration status reports are derived. This includes the maintenance of historical records for use in post-development analysis. The final configuration status accounting function is the data reporting function which includes those reports scheduled on a routine and regular basis as well as unscheduled reports in response to questions concerning the status of the system.

11.2 SOFTWARE CONFIGURATION MANAGEMENT.

Software configuration management is more complex than hardware configuration management, due to the many modifications which can be made to each baseline and the difficulty of ascertaining whether an unapproved change has been incorporated. This necessitates use of a formal audit procedure to insure the integrity of the software product. The RTCA has recommended a method for software configuration management and quality assurance disciplines (reference 6).

Not only must the avionics software be maintained under configuration control, the automatic test equipment and simulator software must also be maintained under configuration control. If they are not, it is impossible that an incompatibility between the avionics line replaceable unit under test and its source data and the line replaceable unit under test and its test software may occur (reference 7).

In the case of software implemented in Read Only Memory (ROM) or Programmable Read Only Memory (PROM), some manufacturers have developed unique configuration control procedures. Memory allocation maps were developed and controlled. The allocation maps were used to track which module of software was stored in PROM or Random Access Memory (RAM). "This map control feature enabled us to release various configurations from one library." (Reference 8). The software in this case included assembly language symbolic modules as well as microcode symbolic modules. "When the instrumentation symbolics were assembled along with the microcode and assembly language modules, absolute locations were identified which were monitored and recorded by the instrumentation hardware." The firmware programs were set up as files on a large mainframe computer in symbolic format. When change requests were initiated, a new symbolic module was created. Support software routines compared the revised module to the library version and modified the revision level on lines that were different. "A listing of the revised element accompanied the change request form. On the form the engineer explained the reason for the changes, as well as the impact on other modes, routines, instrumentation, documentation, and schedule. The change request needed approval from the lead engineer, the engineering manager, the firmware leader, and any other firmware engineer also using the module. Change requests were logged and traced. When approved, a modified module was copied to the library and the old version copied to a recovery file as a precaution against faulty revisions." (Reference 8).

Reference 6 specifically recommends that formal procedures be established to ensure the integrity of the stored data. It states "the masters and the archive duplicates should be created with read-only protection and each copy process should be verified by a bit-by-bit comparison. All media should contain one or more algorithms designed to verify error-free compilation, copying, and loading.

Reference 9 presents an interactive computer program for a configuration management system for software control of configuration management tasks. This interactive computer approach to the configuration management task appears compatible with the desires of the RTCA.

Reference 10 describes an Automated Revision Control System (ARCS) for software. This program automates, controls, documents, and creates visibility for software changes in the various phases of software production, thereby simplifying the process of change documentation.

11.3 SUMMARY.

Configuration management will be required for both hardware and software in all systems. Figure 11-5 provides a synopsis of the principles of configuration control. Even though the methods of software documentation as well as the documents themselves may change (references 8 through 11), their configuration still must be controlled. While there is no one standard procedure to be used for software and hardware configuration control in time, steps are being made toward a more uniform procedure (reference 12). Until this procedure is adopted, the guidelines in reference 6 and this handbook should be observed.

DEFINITION.	<i>Software configuration control is the orchestration of the processes by which the software portion of a system can achieve and maintain visibility through out its journey through the life cycle. It provides the tools (i.e., documentation, procedures, and an organizational body) to control the system implementation as well as any changes to it.</i>
RESPONSIBILITIES	<ol style="list-style-type: none"> 1. To approve, monitor and control the conversion of design objects into system (software) configuration items, and 2. To approve, monitor, and control changes to the system.
	<ul style="list-style-type: none"> • The domain of configuration control includes the satisfaction of both established system requirements and changes in these requirements. • Proper configuration control requires the interaction of buyers, sellers, and users early in the software development process. • Achieving consensus among buyer, seller, and user organizations concerning software design is an integral element of software configuration control. Review of the software design is the mechanism for achieving this consensus. • The CCB, as the focal point of configuration control, must be organized to be responsive to project requirements. In the absence of buyer and/or user awareness of this principle, the seller must fill the void with a set of internal CCBs. In the absence of seller awareness, the buyer must impose the establishment of "The (Integrated) CCB." • Configuration control must begin no later than the start of a system development project. Ideally, the buyer and seller will have established configuration control policies and procedures, as well as a CCB, before a particular project begins. • The review and assessment of the evolving system configuration is appropriately directed and controlled by the CCB. The CCB will invoke other disciplines (e.g., configuration auditing) as required to support such reviews and assessments. • The clearinghouse for proposed revolutionary changes to the system is "The CCB." It will invoke other disciplines as required to analyze the impact of such changes and to maintain records of their implementation status. • The forms of configuration control provide some of the tools necessary to implement an effective SCM program. It is imperative to track and document (via status accounting) both requested changes as well as system problems, and to coordinate the initiation of change with the execution of change. • Configuration management planning is a key element of configuration control. A system CM plan must be prepared at the beginning of the system life cycle, before any baselines are approved. Subsidiary hardware and software CM plans should be developed after approval of the allocated baseline. • Four major areas must be addressed in a CM plan (hardware, software, or system). The first is the organization of the CM body, or CCB, the second is the definition of the set of tools to be applied by CM on the project at hand, the third is the specification of procedures to be followed by CM and the rest of the project staff, the fourth is the budgeting and allocating of resources for the implementation of CM.

FIGURE 11-5. PRINCIPLES OF CONFIGURATION CONTROL (Reference 1)

SECTION 11

11.4 REFERENCES

1. Bersoff, Edward H., Henderson, Vilas D., and Siegel, Stanley G., Software Configuration Management, An Investment In Product Integrity, Prentice-Hall, Englewood Cliffs, NJ, 1980, pp. 209-211.
2. Bersoff, Edward H., Henderson, Vilas D., and Siegel, Stanley G., Software Configuration Management, An Investment In Product Integrity, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980, p. 108.
3. Bersoff, Edward H., Henderson, Vilas D., and Siegel, Stanley G., Software Configuration Management, An Investment In Product Integrity, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980, p. 112.
4. Bersoff, Edward H., Henderson, Vilas D., and Siegel, Stanley G., Software Configuration Management, An Investment In Product Integrity, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980, p. 123.
5. Bersoff, Edward H., Henderson, Vilas D., and Siegel, Stanley G., Software Configuration Management, An Investment In Product Integrity, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980, p. 197.
6. Software Considerations In Airborne Systems and Equipment Certification, Document No. RTCA/DO-178, Radio Technical Commission for Aeronautics, November 1981.
7. Ferrell, John D. R., Software Support Equation of Accountability - A Challenge, Proceedings of the IEEE 1980 National Aerospace and Electronics Conference, pp. 152-153.
8. Sowell, Peggy Jo, Firmware Configuration Management--A Case History, Proceedings of the IEEE 1980 National Aerospace and Electronics Conference, pp. 260-266.
9. Solomon, Eliezer N., and Miller, Francine, Configuration Management System (CMS): A Computerized Approach to Software Configuration Control, Proceedings of the IEEE 1981 National Aerospace and Electronics Conference, pp. 858-865.
10. Duerling, Craig L., An Automated Revision Control System for Software, Ibid, pp. 866-871.
11. Nelson, William P., Some Alternatives to MIL-STD Software Documentation, Ibid, pp. 1141-1146.
12. Berlack, H. Ronald, Hardware and Software Configuration Control History and Current Status, Proceedings of the 1981 RTCA Assembly, Radio Technical Commission for Aeronautics, pp. 167-194.

APPENDIX A

EXAMPLES OF FORMATS FOR DOCUMENTATION

APPENDIX A

SPECIFICATION FORMAT FOR DOCUMENTATION (DRAFT)

EXAMPLE OF FORMATS FOR DOCUMENTATION

Table A-1 is a list of recommended software documents which should be available to the Federal Aviation Administration (FAA) in order to facilitate the certification process. The actual content of each document should be negotiated between the regional FAA certification engineers and the supplier, well in advance of the actual development of the documents.

Most of the documents do not have generic Table of Contents due to the varieties of avionic systems with embedded computer systems and software. However, tables A-2 through A-4 are the recommended Table of Contents for the System Software Development Specification and Product Specification; Subsystem Computer Program Configuration ITEM (CPCI) Development Specification and Product Specification; and the System and Subsystem Software Test Plans.

TABLE A-1. SOFTWARE DOCUMENTATION RECOMMENDED FOR CERTIFICATION
FOR A DIGITAL AVIONIC

1. System Software Configuration Management Plan
 - Documentation Tree
 - Master Index
2. System Software Development Specification
3. System Software Product Specification
4. System Software Development Plan
5. System Software Interface Specification
6. Subsystem Computer Program Configuration Item (CPCI) Development Specification
7. Subsystem Computer Program Configuration Item (CPCI) Product Specification
8. Subsystem Software Code
 - Source Code
 - Object Code
9. System, Subsystem CPCI Software Test Plans
10. System, Subsystem CPCI Software Test Reports
 - CPCI Module Test/Anomaly Report
 - CPCI Integration Test/Anomaly Report
 - System Integration Test/Anomaly Report
 - Acceptance Test/Anomaly Report

TABLE A-2. SYSTEM SOFTWARE DEVELOPMENT SPECIFICATION AND
PRODUCT SPECIFICATION

1. INTRODUCTION

- 1.1 SCOPE
- 1.2 APPLICABLE DOCUMENTS
- 1.3 PURPOSE
- 1.4 SYSTEM OVERVIEW

2. REQUIREMENTS

2.1 OVERALL SOFTWARE REQUIREMENTS

- 2.1.1 Purpose
- 2.1.2 Constraints
- 2.1.3 Performance
- 2.1.4 Criticality of Function
- 2.1.5 Inputs and Outputs - (including constants)
 - 2.1.5.1 Accuracy
 - 2.1.5.2 Update rate
 - 2.1.5.3 Range
 - 2.1.5.4 Gross rate of change
- 2.1.6 Processing - Including logic and mathematical descriptions
- 2.1.7 Responses to Undesired Events

2.2 DETAILED OPERATIONAL REQUIREMENTS

- 2.2.1 Purpose
- 2.2.2 Constraints
- 2.2.3 Performance
- 2.2.4 Criticality of Function
- 2.2.5 Inputs and Outputs - (including constants)
 - 2.2.5.1 Accuracy
 - 2.2.5.2 Update rate
 - 2.2.5.3 Range
 - 2.2.5.4 Gross rate of change
- 2.2.6 Processing - Including logic and mathematical descriptions
- 2.2.7 Responses to Undesired Events

2.3 DETAILED FUNCTIONAL REQUIREMENTS

- 2.3.1 Purpose
- 2.3.2 Constraints
- 2.3.3 Performance
- 2.3.4 Criticality of Function
- 2.3.5 Inputs and Outputs - (including constants)
 - 2.3.5.1 Accuracy
 - 2.3.5.2 Update rate
 - 2.3.5.3 Range
 - 2.3.5.4 Gross rate of change

TABLE A-2. SYSTEM SOFTWARE DEVELOPMENT SPECIFICATION AND
PRODUCT SPECIFICATION (Continued)

- 2.3.6 Processing - Including logic and mathematical descriptions
- 2.3.7 Responses to Undesired Events

2.4 DETAILED SUPPORT SOFTWARE REQUIREMENTS

- 2.4.1 Purpose
- 2.4.2 Constraints
- 2.4.3 Performance
- 2.4.4 Criticality of Function
- 2.4.5 Inputs and Outputs - (including constants)
 - 2.4.5.1 Accuracy
 - 2.4.5.2 Update rate
 - 2.4.5.3 Range
 - 2.4.5.4 Gross rate of change
- 2.4.6 Processing - Including logic and mathematical descriptions
- 2.4.7 Responses to Undesired Events

TABLE A-3. SUBSYSTEM COMPUTER PROGRAM CONFIGURATION ITEM
DEVELOPMENT SPECIFICATION AND PRODUCT SPECIFICATION

1. INTRODUCTION
 - 1.1 SCOPE
 - 1.2 APPLICABLE DOCUMENTS
 - 1.3 PURPOSE
 - 1.4 OVERVIEW (SYSTEM)
2. PROGRAM STRUCTURE AND PARTITIONING
3. PROGRAM CONTROL FLOW WITH TIMING
4. DATA FLOW
5. PROCESSOR ORGANIZATION
 - 5.1 DATE
 - 5.2 PROGRAM
 - 5.3 VALID CPU STATES
 - 5.4 I/O PORTS
6. DESCRIPTIONS OF SUBROUTINES EACH MODULE (BY LEVEL AND FUNCTIONAL AREA)
 - 6.1 PURPOSE
 - 6.2 INPUT
 - 6.3 PROCESSING/ALGORITHM
 - 6.4 OUTPUT
 - 6.5 CPU STATE ALTERATION
 - 6.6 TIMING REQUIREMENT
 - 6.7 FLOW DIAGRAM

TABLE A-4. SYSTEM AND SUBSYSTEM SOFTWARE TEST PLANS

1. INTRODUCTION
 - 1.1 SCOPE
 - 1.2 APPLICABLE DOCUMENTS
 - 1.3 PURPOSE
 - 1.4 SYSTEM OVERVIEW
2. SOFTWARE TEST MANAGEMENT
 - 2.1 GENERAL SOFTWARE TEST OBJECTIVES
 - 2.2 SOFTWARE DEVELOPMENT PLAN (SCHEDULE OF ALL SOFTWARE ACTIVITIES)
 - 2.3 DOCUMENTATION/STORAGE OF TESTS
3. VERIFY SYSTEM SOFTWARE DEVELOPMENT SPECIFICATION AGAINST SYSTEM REQUIREMENTS
 - 3.1 PROCEDURES
 - 3.2 DESIGN REVIEW
 - 3.3 REPORT RESULTS
4. VERIFY SUBSYSTEM COMPUTER PROGRAM CONFIGURATION ITEM DEVELOPMENT SPECIFICATION AGAINST SYSTEM SOFTWARE DEVELOPMENT SPECIFICATION
 - 4.1 PROCEDURES
 - 4.2 DESIGN REVIEW
 - 4.3 REPORT RESULTS
5. TEST MODULES
 - 5.1 GENERAL TEST CRITERIA
 - 5.2 GENERAL TEST PROCEDURE
 - 5.3 BY MODULE
 - 5.3.1 Specific Tests
 - 5.3.2 Test Tools
6. VERIFY CODE OF SUBSYSTEM COMPUTER PROGRAM CONFIGURATION ITEM DEVELOPMENT SPECIFICATIONS
 - 6.1 PROCEDURES
 - 6.2 DESIGN REVIEW
 - 6.3 REPORT RESULTS
7. TEST MODULE INTEGRATION
 - 7.1 GENERAL TEST CRITERIA
 - 7.2 SOFTWARE DEVELOPMENT PLAN (HIERARCHIAL BLOCK DIAGRAM OF MODULE INTERCONNECTIONS)
 - 7.3 GENERAL TEST PROCEDURE (INTEGRATION PLAN)
 - 7.4 BY GROUP

TABLE A-4. SYSTEM AND SUBSYSTEM SOFTWARE TEST PLANS
(Continued)

- 7.4.1 Specific Tests
- 7.4.2 Tools

8. SYSTEM TESTING

- 8.1 GENERAL TEST CRITERIA
- 8.2 SYSTEM REQUIREMENTS VERSUS TEST MATRIX
- 8.3 SYSTEM VERIFICATION

- 8.3.1 Definition
- 8.3.2 Simulation

- 8.3.2.1 Engineering Model
- 8.3.2.2 Prototype

- 8.3.3 Hot Bench (Test Hardware/Software Integration)

- 8.3.3.1 Production Prototype

8.4 SYSTEM VALIDATION

- 8.4.1 Definition
- 8.4.2 Iron Bird
- 8.4.3 Flight Test

9. SOFTWARE MAINTENANCE/REGRESSION TESTING

- 9.1 GENERAL TEST CRITERIA
- 9.2 GENERAL TEST PROCEDURE
- 9.3 BY MODULE AND GROUP REQUIRING RETEST

- 9.3.1 Specific Tests

APPENDIX B

RELIABILITY ANALYSIS MODELS, FAULT TREES FAILURE MODES AND EFFECTS ANALYSIS SYNOPSIS

APPENDIX B

RELIABILITY ANALYSIS MODELS, FAULT TREES, FAILURE MODES AND
EFFECTS ANALYSIS SYNOPSES

APPENDIX B

RELIABILITY ANALYSIS MODELS, FAULT TREES, FAILURE MODES AND EFFECTS ANALYSIS SYNOPSES

ARIES (AUTOMATED RELIABILITY INTERACTIVE ESTIMATION SYSTEM)

ARIES is an interactive computer program based on a unified method of reliability analysis. Developed in 1976 at UCLA, it uses a Markov approach to model systems of independent homogenous subsystems with various types of hardware redundancy possible. The theory and development of ARIES are well documented (references B-1 through B-4). Thorough user's guides (references B-5 and B-6) are available. ARIES 81 has been formally validated by comparison with ARIES 76 results by UCLA. ARIES 81 is written in the C computer language and runs under the UNIX operating system on a VAX IV/780 computer. In addition to system reliability as a function of time, ARIES computes the maximum mission time for which reliability is at least a user-specified value, the relative contribution of a component to system unreliability, and the reliability improvement factor.

ARIES can handle a wide variety of hardware redundancy approaches. All modules have constant hazard rates. The modeling of coverage and transient faults is quite adequate, given the current state of knowledge about those factors. Different coverage factors can be used for recovery from active and spare module failures. Transients are modeled with constant hazard rates and mean durations. Recovery is assumed to be instantaneous, which may or may not be a good assumption. A system must be treated as a series of independent subsystems, each of which is composed of like modules. A spare module may have a different failure rate than active modules. ARIES cannot handle the situation where a failure of one module causes the functional loss of another module in a different subsystem.

The user of ARIES must have an understanding of basic reliability theory and Markov models, and detailed knowledge of the system to be modeled. The interactive prompting messages appear quite adequate for the data input function. ARIES takes advantage of subsystem independence and file storage of subsystems and their eigenvalues to reduce the machine time required to execute a run.

CARE II

CARE II was developed in 1974 by Raytheon to study the role of coverage in fault-tolerant systems. In addition to system reliability as a function of time, CARE II computes the contributions of fault detection, isolation, and recovery features to the success of the coverage function. Reference B-7 provides a user's guide, sample runs, flow charts, program source listings, and detailed technical discussion of the model. Several test runs of CARE II in reference B-7 were compared to hand-calculated results. Raytheon has used the model to evaluate fault-tolerant processors and computers for several sponsors, but it is not known if any validity checks were performed in those efforts. CARE II was written for a 60-bit CDC series 6000 computer using RUN FORTRAN under the KONOS 2.1 or SCOPE 3.0 operating systems. The basic version uses 100,000 words of memory. An enhanced version which plots system reliability as a function of time uses 130,000 words.

CARE II was designed to study in detail the effects of coverage on system reliability. This orientation is manifested by detailed input data required to describe the elements of coverage for the system to be modeled. Fault detection and isolation parameters are available for transient faults, transient faults mistaken for permanent faults, active module permanent faults, and standby module permanent faults. Timed and untimed fault detection capability can be represented. Recovered modules are allowed to reenter the pool of spares with operational mode reduction. Spare modules failure-rates may be identical to or different from active module failure-rates. Coverage factors may be dependent upon the nature of the fault and its location in the system. CARE II allows up to eight independent stages (i.e., subsystems).

Only two operating modes for the system are allowed by CARE II. This may be a serious restriction for digital flight control and avionics systems which can exhibit many stages of degradation. For example, a triplex-duplex-simplex system cannot be modeled with CARE II.

The user of CARE II must have detailed knowledge of the system being modeled to properly use the model. The model appears reasonable to execute. It allows for multiple cases per run, which is an aid to sensitivity analysis. The quality of the available documentation should aid the analyst in learning to use CARE II.

CARSRA

CARSRA was developed in 1976 by Boeing as part of a study of reconfigurable digital flight control systems (reference B-8). The system to be modeled is partitioned into stages. A stage is a group of identical, redundant modules and a module is a set of elements performing a specified function (e.g., radio altimeter, processor). Each stage is represented by a Markov model with up to nine states. The various types of hardware faults (permanent, transient, and latent) and coverage are modeled by specifying the state transition rates. For some stages, the failure of one of its modules causes the loss of function of another module in a different stage. CARSRA handles these dependencies between stages using a set of mutually exclusive and exhaustive conditional probabilities. Functional redundancy, which is a trait of some integrated flight control systems, can be modeled to a limited extent by tabulating the stage combinations which result in system success.

CARSRA computes system reliability as a function of time where system success considers functional redundancy, module redundancy, and dependencies between stages. CARSRA also computes functional readiness, which is the probability of having at least a minimum, user-specified set of system functions available at a given point in the mission. The functional readiness feature can be used to assess the probability of experiencing a certain degradation in the first part of a mission and then successfully completing the mission. For example, CARSRA can compute the probability of being able to initiate autoland at time t , and then landing safely at time t_2 .

Thorough documentation of CARSRA is available (reference B-8). It includes model theory, user's guide, an example, and subroutine descriptions. The program was written for CDC computers and consists of approximately 800 FORTRAN IV statements.

No formal validation of CARSRA has been performed. CARSRA results have been compared with hand computations and results from other models by the model authors and several users. Boeing applied CARSRA to the flight control system of the Boeing 747 aircraft (reference B-9). Redundancy, coverage, dependencies, and permanent faults were modeled. Another Boeing model which uses tabulation of success paths was also applied. The results of the two models agreed to two digits for probabilities on the order of 10^{-8} . Litton Industries used CARSRA to study the sensor portion of flight control systems with shared instruments (reference B-10). The CARSRA results compared satisfactorily to the results of hand computations.

A Markov model is constructed for each stage in the system. It allows flexibility for modeling permanent, transient, and latent faults; coverage; and module redundancy. The approach to capturing dependencies between stages limits state transitions to one direction. This limitation restricts the flexibility of CARSRA to represent transient faults. Transient faults are modeled using the proportion of transients which are viewed by the system as permanent and the probability of a second fault during the presence of a transient. CARSRA is limited to fifty stages, which should not be a problem for systems of moderate size.

The user of CARSRA must be familiar with the basics of Markov models. The user must understand the behavior of transient faults and fault recovery methods in the context of the system being modeled to estimate the state transition rates. Creation of the "system dependency tree" (a logical structure for representing the functional dependencies between stages) also requires the user to be very familiar with the system. Computer run times on the order of 5 to 30 seconds have been reported.

CAST

CAST was developed in 1974 by Ultrasystems as part of a study performed for NASA-Langley Research Center (reference B-11). The objective of the overall study was evaluation of concepts and data to aid in the design of fault-tolerant computers. The model itself was not a deliverable item. The objective of CAST was to help characterize the effects of transient faults and possible recovery schemes on system reliability.

CAST uses Monte Carlo simulation to estimate coverage parameters which are then input to an analytic Markov model to compute system reliability. The simulation uses input data describing the fault detection features, recovery features, system failure criteria, and the executive structure. By simulating a large number of transients, estimates are derived for factors such as transient leakage, recoverability, and detectability. The final outputs of CAST are system failure probability for a specified mission time and fault coverage statistics from the simulation.

A major drawback of CAST is the lack of available documentation. While reference B-11 describes the nature and features of the model, no user's manual, flowcharts, or program listings are included. CAST has reportedly been applied to a large number of computer configurations, but no validation has been performed. The model was programmed for the CDC-6600 computer in FORTRAN IV.

CAST appears to be limited to system configurations of up to five identical computers. Treatment of sensors and servos is unclear. The description of the model indicates there are the same number of identical groups of input sensors

as there are computers. A sensor group may be dedicated to a particular computer or nondedicated. While CAST can model transient faults and recovery mechanisms in detail, it cannot handle the nonidentical input features and fault interdependencies that exist in digital flight control systems.

The description of CAST (reference B-11) indicates that a user should understand Monte Carlo simulation and statistical estimation. In addition, detailed knowledge of the system being modeled is required. Computer time requirements could not be estimated from the available documentation. Machine time could be considerable in using the simulation to obtain acceptable confidence levels for the coverage statistics.

CARE III

CARE III has been developed by Raytheon for NASA-Langley Research Center with the objective of overcoming some of the limitations of CARE II and other models. The design specifications for CARE III were based on the types of system configurations, faults, and recovery schemes associated with current and future designs for digital flight control systems. As a result, the goals for CARE III are more ambitious than the goals for the other models. CARE III will be the most powerful of the reviewed models if its objectives are met.

No formal validation of CARE III has been performed, but a number of activities which contribute to validation have been done or are planned. Major algorithms used in the model were validated during the development process. Raytheon has applied CARE III to versions of the SIFT and FTMP computer systems. The CARE III results compared quite well to the reliability estimates obtained by the developers of the systems. A "peer review" of CARE III assessed the modeling approach, the numerical method, and issues for further investigation. NASA-Langley is planning a contract research effort to develop confidence in CARE III by deriving the component models, investigating assumptions and approximations, and applying the model to a set of test cases.

Thorough documentation is available for both the development and the implementation of CARE III. The model was written in FORTRAN Extended 4 language for execution on CDC computers. Interactive capability for creation of input data files is included. Attention has been given to the user interface with the model, but the effectiveness of this effort is not yet known.

CARE III can handle up to 40 stages, multiple modes of operation for each set of interdependent stages, hazard rates which vary as a function of time, and nonunity dormancy factors. Some dependencies between stages can be modeled. The coverage model is separate from the system reliability model. This is a rational decomposition since the reliability is concerned with rate events (i.e., failures) and coverage is concerned with rapid events (i.e., detection, isolation, recovery) given a fault has occurred. This separation allows the model user to concentrate on each area relatively independently.

CARE III appears to require approximately the same user skills and knowledge as the other models when applied to the same problem. Computer run times are expected to be longer for CARE III than for the other models. CARE III uses a recursive technique for reliability evaluation (reference B-12) and such techniques generally require more time than closed-form algorithms. The run times are still expected to be reasonable.

APPENDIX B

REFERENCES

- B-1. Ng, Y. W., Reliability Modeling and Analysis for Fault-Tolerant Computers, National Science Foundation, Report No. NSF-MCS-7203633-76981; Ph.D. Dissertation, University of California, Los Angeles, Report No. UCLA-ENC-7698, September 1976.
- B-2. Ng, Y. W., and Avizienis, A., ARIES - An Automated Reliability Estimation System for Redundant Digital Structures, Proceedings of the 1977 Annual Reliability and Maintainability Symposium, Philadelphia, PA, January 1977, Pages 108-113.
- B-3. Ng, Y. W., and Avizienis, A., A Reliability Model for Gracefully Degrading and Repairable Fault-Tolerant Systems, Proc. FTC-7, Los Angeles, 1977, Pages 22-28.
- B-4. Makam, Srinivas V., and Avizienis, A., ARIES 81: A Reliability and Life-Cycle Evaluation Tool for Fault-Tolerant Systems, Proceedings of FTSC-12, June 1982.
- B-5. Ng, Y. W., and Avizienis, A., ARIES 76 User's Guide, National Science Foundation, Report No. NSF-MCS-7203633-78944, University of California, Los Angeles Report No. UCLA-ENG-7894, December 1978.
- B-6. Makam, Srinivas, Avizienis, Algridas, Grusas, and Gintaras, ARIES 81 Users' Guide, Computer Science Department, School of Engineering and Applied Science, UCLA, June 1982.
- B-7. Reliability Model Derivation of a Fault-Tolerant, Dual, Spare-Switching Digital Computer System, Prepared for NASA-Langley Research Center by Raytheon Company, Contract NAS1-12668, March 1974.
- B-8. Bjurman, B. E., et al, Airborne Advanced Reconfigurable Computer System (ARCS), NASA CR-145024, Prepared for NASA-Langley Research Center, NASA by Boeing Commercial Airplane Company under contract NAS1-13654, August 1976.
- B-9. Private Communications with Roger H. Edwards, Boeing Commercial Airplane Company, Seattle, Washington.
- B-10. Private Communications with Bob Ebner, Litton Industries, Los Angeles, California.
- B-11. Conn, R. B., et al, Definition and Trade-Off Study of Reconfigurable Airborne Digital Computer System Organizations, prepared for NASA-Langley Research Center by Ultrasystems, Incorporated, Contract NAS1-12793, November 1974.
- B-12. Stiffler, J. J., et al. CARE III Final Report. Phase I. NASA CR-159122. November 1979.

APPENDIX C
FAULT TREE EXAMPLE

APPENDIX C
FAULT TREE EXAMPLE

APPENDIX C

FAULT TREE EXAMPLE

Purpose

The purpose of this example is to demonstrate the application and use of fault trees in system reliability analysis. Discussion of the theory of fault trees is not provided since the literature is replete with tutorial and advanced material.

The Problem

The example selected for analysis is loss of bus control in the HH-65A helicopter avionics system. This avionics system is described in the FMS Subsystem Specification (reference C-1), which is the specification document for the system. A block diagram of the system is reproduced in figure C-1. Bus control is described in Section 3.4.1 (pages 57 to 66) of reference C-1.

Fault Tree Construction

The top event of the fault tree is "Loss of Bus Control." Five events which could cause the top event to occur were identified. They are:

- Loss of both buses.

- Loss of the clock function.

- Failure of the MCU to give up bus control, given the clock function is operating correctly.

- Failure of the SCUs to accept bus control from the MCU.

- An RT generates uncontrolled outputs.

Since any one of these events can cause loss of bus control, they are connected by a logical OR gate.

Each of the five events in the above list was decomposed into its contributing events. This process was continued until events whose probabilities of occurrence could be estimated were identified. In order to ensure that all bottom level, or "basic," events which could cause loss of bus control were accounted for, a failure mode and effects analysis (FMEA) was performed for failures of the components associated with bus control.

Figure C-2 presents the fault tree for loss of bus control. As noted above, the fault tree is based on the description of bus implementation in the avionics system specification. Paragraph 3.4.1.4 of reference C-1 specifies the use of fail-safe design techniques to ensure bus operational integrity. The fault tree accounts for the techniques specifically discussed in the text. However, the actual implementation of bus control may utilize additional techniques or variations which would require changes in the fault tree.

Probability Computations

The probability of the top event, loss of bus control, is computed using a "bottom-up" procedure. The component failure rate data are used to compute the probability of each basic event. A mission time of 2 hours is assumed (i.e., $t = 2$). The probabilities of the basic events are combined in accordance with the logical gates to determine the probability of the next higher event.

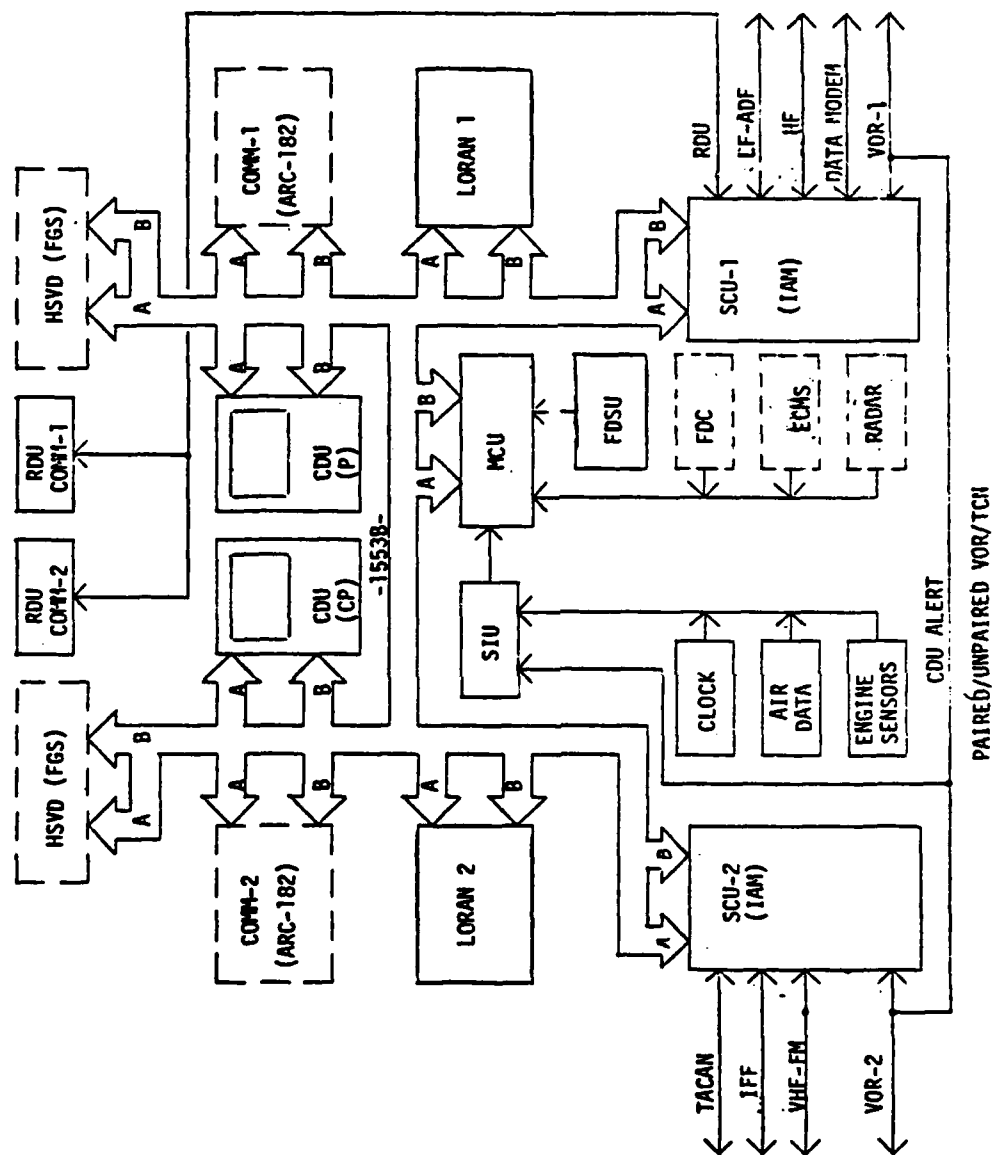


FIGURE C-1. FMS HARDWARE FUNCTIONAL BLOCK DIAGRAM

ROCKWELL INTERNATIONAL CORPORATION COLLINS GROUPS DALLAS, TEX 75207 NEWPORT BEACH, CALIF 92663 CEDAR RAPIDS, IA 52406			
PREP	SIZE A	FSCM 13499	DWG NO. 642-8739-001
CHK	SCALE	SHEET 12	REV C LTR

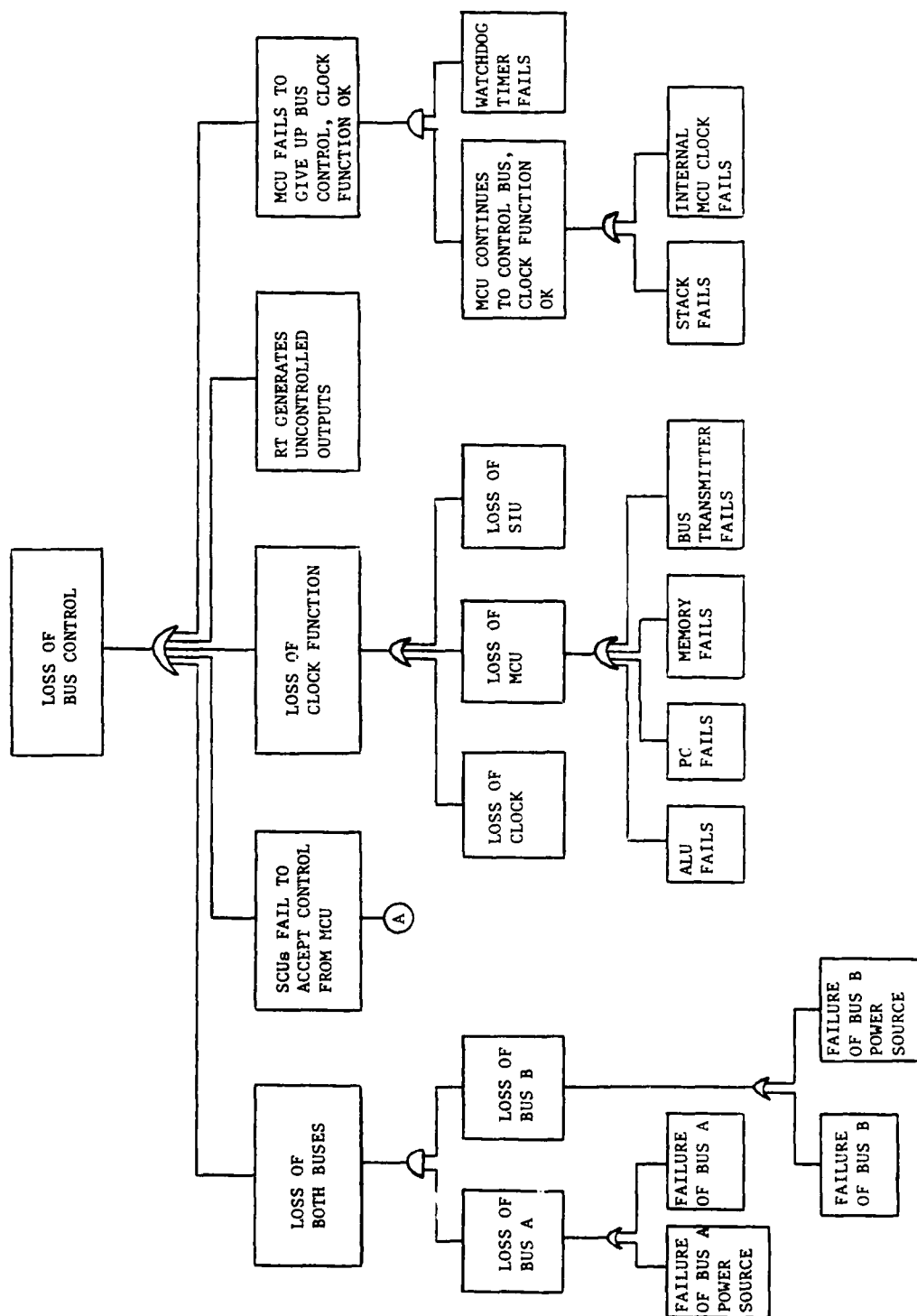


FIGURE C-2. FAULT TREE FOR LOSS OF BUS CONTROL (1 of 2 Sheets)

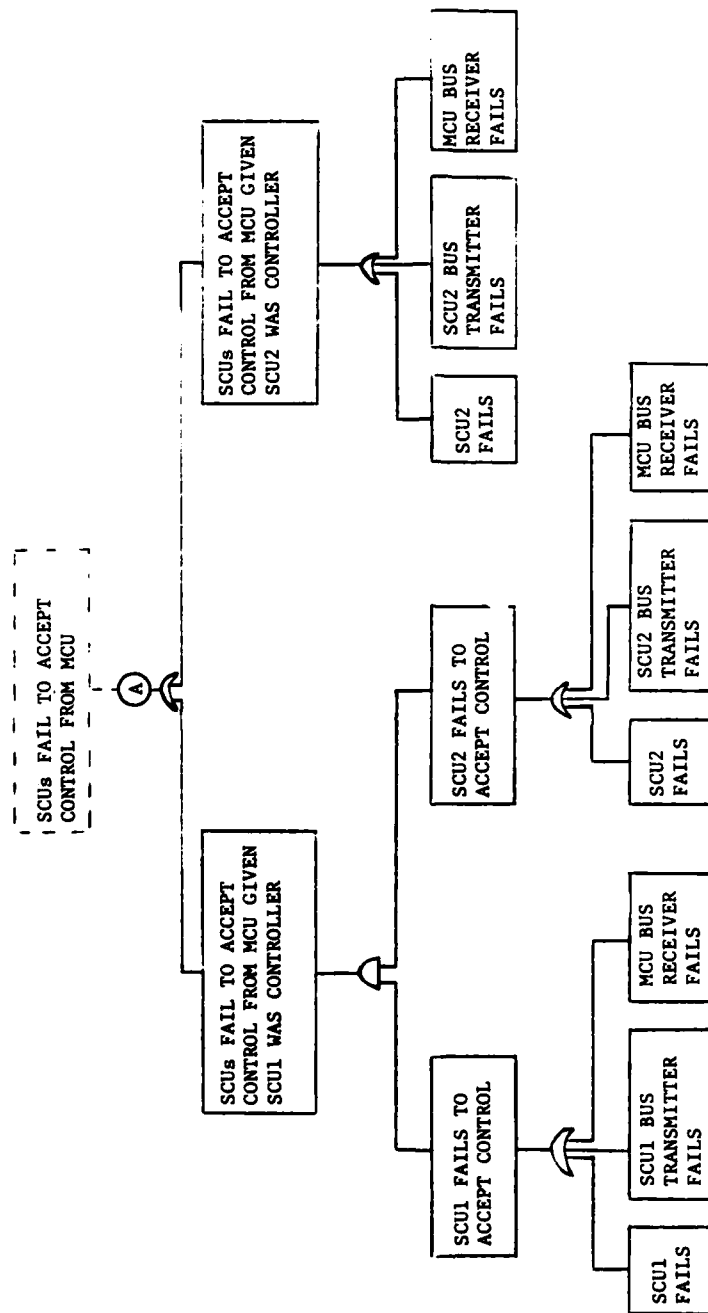


FIGURE C-2. FAULT TREE FOR LOSS OF BUS CONTROL (2 of 2 Sheets)

OR Gate. Suppose event E can be caused by either event X1 or event X2 (or both); i.e., events X1 and X2 are connected by an OR gate. The probability of event E is then computed by:

$$\begin{aligned} P(E) &= 1 - [1 - P(X1)][1 - P(X2)] \\ &= P(X1) + P(X2) - P(X1) P(X2) \end{aligned}$$

If $P(X1)$ and $P(X2)$ are both small, then a good approximation for $P(E)$ is:

$$P(E) = P(X1) + P(X2)$$

AND Gate. Suppose event E will occur only if both events X1 and X2 occur (i.e., events X1 and X2 are connected by an AND gate). The probability of event E is computed by:

$$P(E) = P(X1) P(X2)$$

Basic Event Probabilities. Table C-1 presents the basic event probabilities for the fault tree. The failure data in the table are fictitious and are used for demonstration purposes only. For each component, the failure probability is computed by:

$$P(\text{failure}) = 1 - \exp(-\lambda t)$$

where λ = failure rate = $1/\text{BF}$ and t = mission time.

TABLE C-1. BASIC EVENT PROBABILITIES

	Mean Time Between Failures, Hrs	λ , Failures/ 10^6Hr	Failure Probability, $t = 2 \text{ hrs}$
SCU	4,000	250	0.000499875
SCU Bus Transmitter	5,000	200	0.000399920
SCU Bus Receiver	6,000	167	0.000333278
Watchdog Timer	8,000	125	0.000249969
MCU ALU	30,000	33	0.000066665
MCU Program Counter	40,000	25	0.000049999
MCU Memory	50,000	20	0.000039999
MCU Bus Transmitter	5,000	200	0.000399920
MCU Receiver	6,000	167	0.000333278
MCU Internal Clock	50,000	20	0.000039999
MCU Stack	1,000,000	1	0.000002000
Clock	10,000	100	0.000199980
RT	7,000	143	0.000285764
SIU	20,000	50	0.000099995
Power Source	5,000	200	0.000399920
Bus	20,000	50	0.000099995

Example Computation. As an example computation, consider the sub-tree whose top event is "MCU fails to give up bus control, clock function OK." Beginning at the lowest level events, we have

$$P(\text{stack fails}) = 0.000002000$$

$$P(\text{internal MCU clock fails}) = 0.000039999 \quad .$$

Applying the formula for an OR gate:

$$\begin{aligned} P(\text{MCU fails to control bus,} \\ \text{clock function OK}) &= 1 - (1 - 0.000002000) (1 - 0.000039999) \\ &= 0.000041999 \quad . \end{aligned}$$

This result is combined with the $P(\text{watchdog timer fails})$ by an AND gate to determine the probability of the top event of the sub-tree:

$$\begin{aligned} P(\text{MCU fails to give up bus control,} \\ \text{clock function OK}) &= (0.000041999) (0.000249968) \\ &= (0.000000010) \quad . \end{aligned}$$

Results. Figure C-3 presents the complete fault tree with the associated event probabilities.

Fault Tree Applications

The results of the fault tree computations can be used to identify the "failure drivers"--those components (or events) which make the largest contributions to the top event. For the first level under the top event in figure C-3, two events ("SCUs fail to accept control from MCU" and "loss of clock function") account for almost 90 percent of the probability of loss of bus control. Efforts to improve the reliability of bus control therefore should be focused on these two events.

The effects of changes in component failure rates on the top event can be readily computed. Such sensitivity analysis is an input to determination of cost-effective improvements.

REFERENCE

1. HH-65A, Avionics System Specifications, AHC 366-S-007, December 21, 1979, for the U.S. Coast Guard.

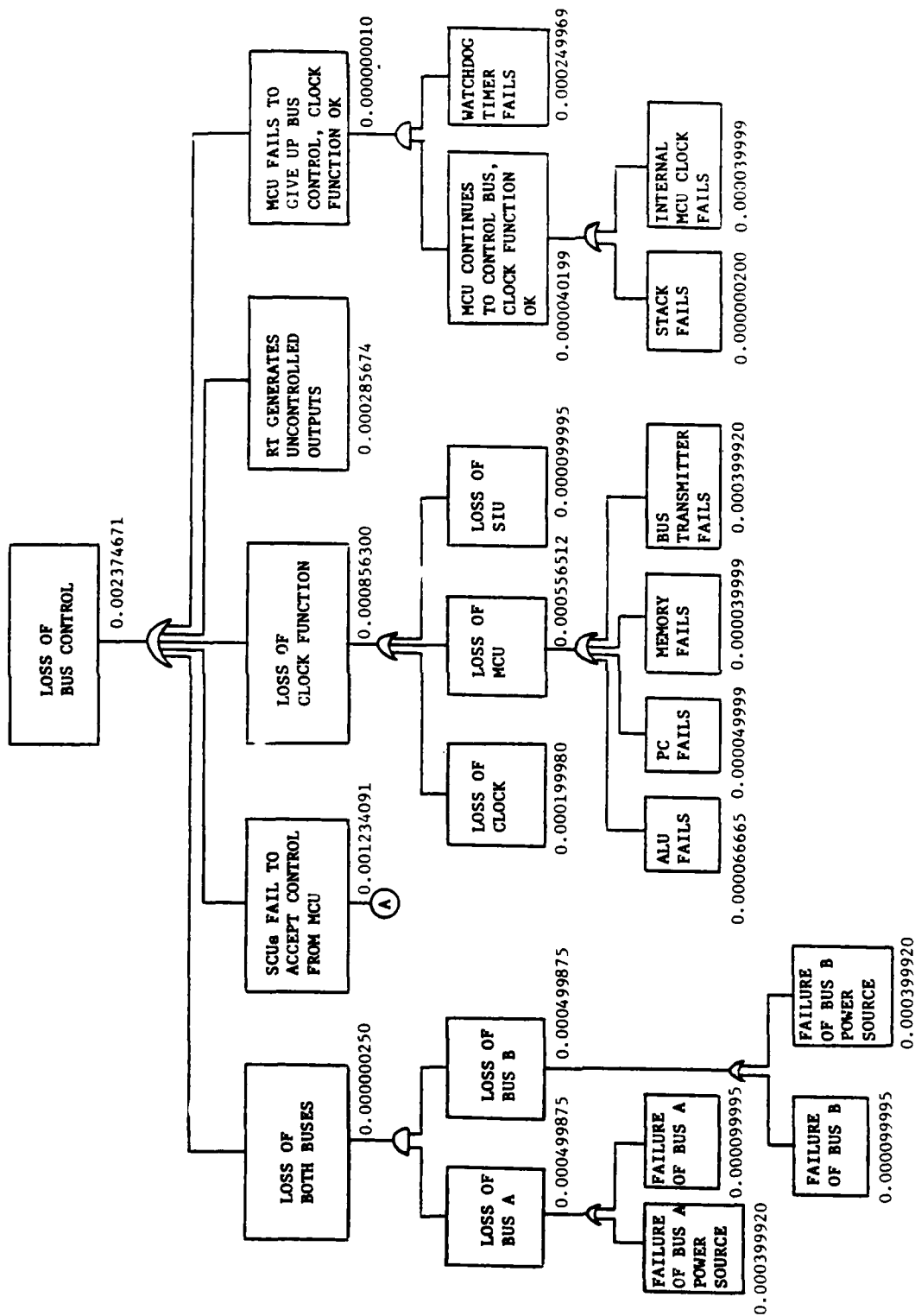


FIGURE C-3. FAULT TREE FOR LOSS OF BUS CONTROL WITH PROBABILITY (1 of 2 Sheets)

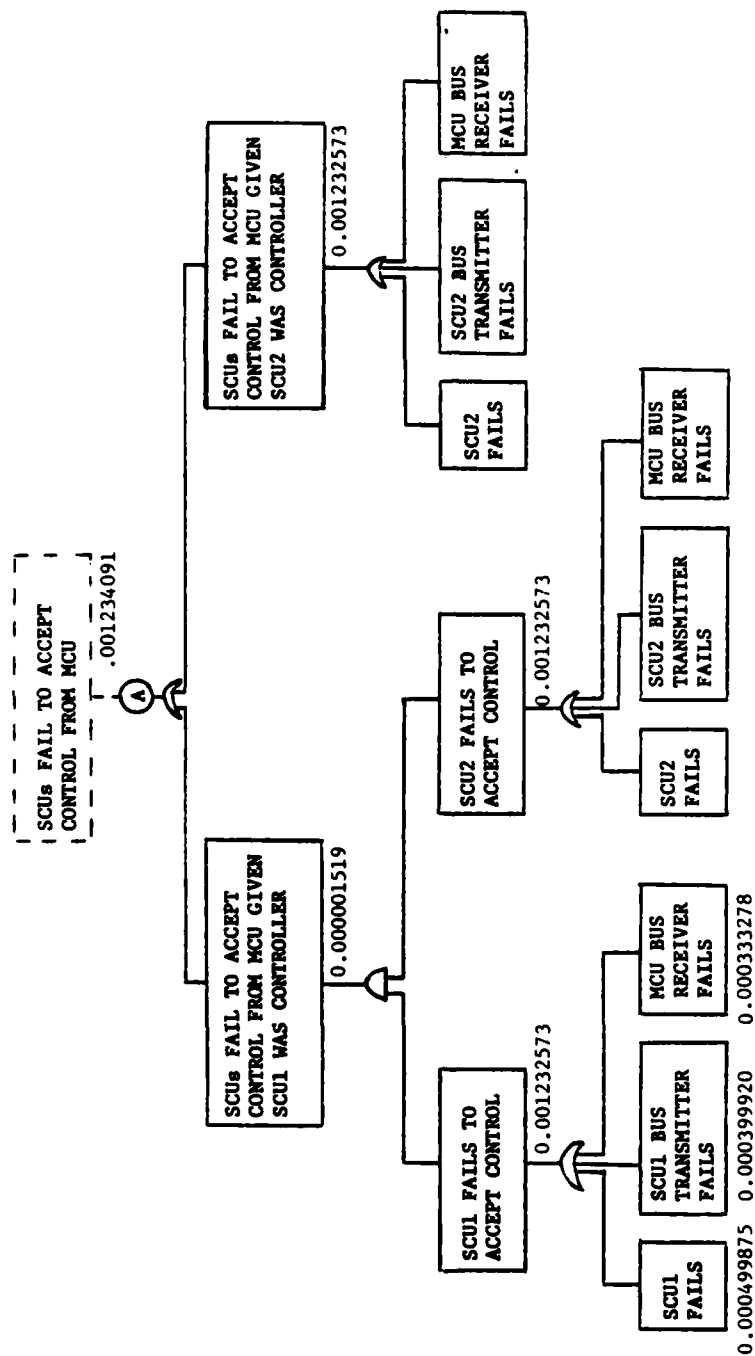


FIGURE C-3. FAULT TREE FOR LOSS OF BUS CONTROL WITH PROBABILITY (2 of 2 Sheets)

APPENDIX D

**ADVISORY CIRCULAR AC: 25:1309-1
SYSTEM DESIGN ANALYSIS**

APPENDIX D

ADVISORY CIRCULAR AC: 25:1309-1
SYSTEM DESIGN ANALYSIS



U.S. Department
of Transportation
**Federal Aviation
Administration**

Advisory Circular

Subject: SYSTEM DESIGN ANALYSIS

Date: 9/7/82
Initiated by: ANM-110

AC No: 25.1309-1
Change:

1. PURPOSE. This advisory circular provides guidance material for acceptable means, but not the only means, of demonstrating compliance with the requirements of Part 25 of the Federal Aviation Regulations which includes probabilistic terms, as introduced by Amendment 25-23, for airplane equipment, systems, and installations.

2. REFERENCE REGULATION. Section 25.1309 of the Federal Aviation Regulations, as amended through Amendment 25-41.

3. BACKGROUND.

a. For a number of years, aircraft systems were evaluated by the Federal Aviation Administration to the "single fault" criteria contained in § 4b.606 of the Civil Air Regulations, recodified and later amended as § 25.1309 of the Federal Aviation Regulations. The term "single fault" was a misnomer because additional cases of the hidden fault and the consequential fault also had to be considered (§ 4b.606-1 of the Civil Aeronautics Manual). With the development of more complex systems and the increasing criticality of those systems, the Federal Aviation Administration revised the rules in 1970 to require consideration of single and multiple faults in the system under study. The consequences of faults in separate systems which perform different functions are also to be considered if the simultaneous loss of functions performed by these systems creates a hazard to the airplane. Because of the growth in airplane system complexity, it is difficult in certain cases to make a responsible engineering judgment regarding the effects of certain system failures based on conventional analysis, tests, and historical data. However, the need for making a valid judgment has increased with the increasing criticality of certain systems.

9/7/82

b. To better understand the effects of complex airplane system failures, it may be desirable to use analytical techniques which can assist in identifying failure conditions and their potential consequences. This advisory circular identifies various analytical approaches, both qualitative and quantitative, which may be used to assist airplane manufacturers and FAA personnel in determining compliance with the referenced regulation and provide guidance for determining when, or if, a particular analysis should be conducted. Numerical values are assigned to the probabilistic terms included in the referenced regulation for use in those cases where the effects of system faults are examined by quantitative methods of analysis.

c. A finding of compliance with the requirements of FAR 25.1309 is based on the technical judgment of FAA pilots and engineers. The structured methods of analysis described by this advisory circular are intended to assist FAA personnel in finding compliance with the requirements in those cases where a design review cannot readily determine the impact of failures on the safety of the airplane. These analytical tools are intended to supplement, but not replace, the judgment of the FAA certification personnel.

4. DISCUSSION.

a. Section 25.1309 of Part 25 of the Federal Aviation Regulations, subsequent to Amendment 25-23, requires substantiation by analysis and, where necessary, by appropriate ground, flight, or simulator tests that the probability of a failure condition is expected to remain within limits which are related to the consequence of the failure condition. The requirements in the referenced regulation are intended to assure an orderly and thorough evaluation of systems considered separately and in relation to other systems. It is important to recognize that some systems (functions) have conventionally received such an evaluation to show compliance with other specific regulations or special conditions and thereby may be shown to meet the intent of FAR 25.1309 without a need for extensive additional analyses.

b. The probability of the occurrence of a failure condition may be considered within three classifications: probable, improbable, and extremely improbable. These classifications may be related to failure conditions which have increasingly more severe impact on safety. Airplane functions may be divided in the following manner:

(1) NON-ESSENTIAL--Functions whose failures would not contribute to or cause a failure condition which would significantly impact the safety of the airplane or the ability of the flight crew to cope with adverse operating conditions. Airplane conditions which result from improper accomplishment or loss of non-essential functions may be probable.

(2) ESSENTIAL--Functions whose failures would contribute to or cause a failure condition which would significantly impact the safety of the airplane or the ability of the flight crew to cope with adverse operating conditions. Failure conditions which result from improper accomplishment or loss of essential functions must be improbable.

(3) CRITICAL--Functions whose failure would contribute to or cause a failure condition which would prevent the continued safe flight and landing of the airplane. Failure conditions which result from improper accomplishment or loss of critical functions must be extremely improbable.

c. In order to show compliance with FAR 25.1309(b), FAR 25.1309(d) requires an analysis which must consider:

(1) Possible modes of failure, including malfunctions and damage from external sources.

(2) The probability of multiple failures and undetected failures.

(3) The resulting effects on the airplane and occupants, considering the stage of flight and operating conditions, and

(4) The crew warning cues, corrective action required, and the capability of detecting faults.

d. An analysis to identify failure conditions should be qualitative. An assessment of the probability of a failure condition can be qualitative or quantitative. An analysis may range from a simple report which interprets test results or presents a comparison between two similar systems to a fault/failure analysis which may (or may not) include numerical probability data. An analysis may make use of previous service experience from comparable installations in other airplanes.

e. The depth of this analysis will vary, depending on the design complexity and type of functions performed by the system being analyzed. Section 6 of this advisory circular identifies various analytical approaches and provides guidelines for determining when each should be used.

5. DEFINITIONS. For the purpose of conducting or evaluating an analysis, the following terms and numerical values should apply:

a. CONTINUED SAFE FLIGHT AND LANDING--This phrase is used in the regulation to require that an airplane be capable of continued controlled flight and landing, possibly using emergency procedures and without exceptional pilot skill or strength, after any failure condition which has not been shown to be extremely improbable. There may be failure conditions which are not extremely improbable for which it is necessary to assure that continued safe flight and landing is possible by appropriate analysis and/or tests.

b. DEDUCTIVE--The term used to describe those analytical approaches involving the reasoning from a defined unwanted event or premise to the causative factors of that event or premise by means of a logical methodology (the "top-down" or "how could it happen" approach). A deductive approach will postulate a particular failure condition and attempt to determine what system and equipment failure modes, errors, and/or environmental conditions will contribute to the failure condition.

c. ERROR--A mistake in specification, design, production, maintenance, or operation which causes an undesired performance of a function.

d. EVENT--An occurrence which causes a change of state. NOTE: The Regulatory Authorities of some countries use a more specific definition.

e. EXPOSURE TIME--The period (in clock time or cycles) during which a system, subsystem, unit or part is exposed to failure, measured from when it was last verified functioning to when its proper performance is or may be required.

f. FAILURE--The inability of a system, subsystem, unit or part to perform within previously specified limits. Note that some failures may have no effect on the capability of the airplane and therefore are not failure conditions.

g. FAILURE ANALYSIS--The logical, systematic examination of a system, subsystem, unit or part, to identify and analyze the probability, causes, and consequences of potential and real failures.

h. FAILURE CONDITION--A consequential airplane state which has an impact on the functional capability of the airplane or the ability of the crew to cope with adverse operating conditions, or which would prevent continued safe flight and landing. NOTE: A failure condition can result from the occurrence of a specific single event or a combination of related faults, failures, errors, operating conditions or environments. Postulated failure conditions are assessed for their impact on safety and assigned an appropriate probability classification. A defined failure condition provides the criteria for classifying system functions as non-essential, essential or critical and for showing compliance with 25.1309(b) in accordance with this advisory circular.

i. FAILURE EFFECT(S)--The consequence(s) of a failure mode on the system, subsystem, unit or part's operation, function, or status.

j. FAILURE MODE--The manner in which a system, subsystem, unit, part or function can fail.

k. FAULT--An undesired anomaly in the functional operation of a system, subsystem, unit or part.

l. FAULT TREE ANALYSIS--A top down deductive analysis identifying the conditions and functional failures necessary to cause a defined failure condition. The fault tree, when fully developed, may be mathematically evaluated to establish the probability of the ultimate failure condition occurring as a function of the estimated probabilities of identified contributory events.

m. FLIGHT TIME (Block Time)--The time from the moment the aircraft first moves under its own power for the purpose of flight until the moment it comes to rest at the next point of landing.

n. PROBABILITY CLASSIFICATIONS--Three probability classifications are defined below. Quantitative ranges are also provided as a common point of reference if numerical probabilities are used in assessing compliance with FAR 25.1309 or other applicable regulations. The quantitative ranges given for these classifications represent goals and are considered to overlap due to the inexact nature of probability estimates. When assessing the acceptability of a failure condition using a quantitative analysis, the numerical ranges given below should normally be interpreted to be the allowable risk for an hour of flight time based on a flight of mean duration for the airplane type. However, when assessing a function which is used only at a specific time during a flight, the probability of the failure condition should be calculated for the specific time period and expressed as the risk for the flight condition; takeoff, landing, etc., as appropriate.

(1) PROBABLE--Probable events may be expected to occur several times during the operational life of each airplane. A probability on the order of 1×10^{-5} or greater.

(2) IMPROBABLE--Improbable events are not expected to occur during the total operational life of a random single airplane of a particular type, but may occur during the total operational life of all airplanes of a particular type. A probability on the order of 1×10^{-5} or less.

(3) EXTREMELY IMPROBABLE--Extremely improbable events are so unlikely that they need not be considered to ever occur, unless engineering judgment would require their consideration. A probability on the order of 1×10^{-9} or less.

NOTES: (a) If a quantitative analysis is used to help show compliance with Federal Aviation Regulations for equipment which is installed and required only for a specific operating condition for which the airplane is thereby approved, credit may not be taken for the fact that the operating condition does not always exist. Except for this limitation, appropriate statistical randomness of environmental or operational conditions may be considered in the analysis.

(However, the applicant should obtain prior concurrence of the FAA when including such conditions in the analysis.) (b) The three probability terms defined in paragraph 5n above are intended to relate to an identified failure condition resulting from or contributed to by the improper operation or loss of a function or functions. These terms do not define the reliability of specific components or systems. (c) The range of numerical values assigned to each of the terms is intended to minimize differences in the interpretation of what these terms mean when used in § 25.1309 of the Federal Aviation Regulations. It is important to realize that these terms and others such as "reliable," "unlikely," and "remote" are used throughout the Federal Aviation Regulations. In many cases, these other terms were used prior to Amendment 25-23. Careful

judgement is necessary when interpreting the intent of any regulation using such terms. In all cases, the effect of the given failure conditions should be considered.

o. FUNCTION--Each particular purpose performed by a system, subsystem, unit or part.

p. INDUCTIVE--The term used to describe those analytical approaches involving the systematic evaluation of the defined parts or elements of a given system or subsystem to determine specific characteristics of interest (the "bottom-up," or "what happens if" approach). An inductive approach will assume an initiating event and attempt to determine the corresponding effect on the overall system.

q. HIDDEN FAILURE--A failure that is not inherently revealed at the time it occurs.

r. QUALITATIVE--The term used to describe those analytical approaches which are oriented toward relative, nonmeasurable or non-numerical and subjective values.

s. QUANTITATIVE--The term used to describe those analytical approaches which are oriented toward the use of numbers to express a measurable quantity.

t. REDUNDANCY--The existence of more than one independent means of accomplishing a given function.

u. RELIABILITY--The probability that a system, subsystem, unit or part will perform its intended function for a specified interval under stated operational and environmental conditions.

6. ACCEPTABLE TECHNIQUES.

a. The first step in determining compliance with FAR 25.1309(b) should be to determine the criticality of the system or installation to be certificated. This analysis may be conducted using service experience, engineering, or operational judgment, or by using a top-down deductive qualitative analysis which examines each function performed by the system. The analysis should determine the criticality of each system function, i.e., either non-essential, essential, or critical. Each system function should be analyzed with respect to functions performed by other aircraft systems. This is necessary because the loss of different but related functions provided by separate systems may affect the criticality category assigned to a particular system. This type of analysis, variously referred to as a preliminary hazard analysis, criticality categorization, or criticality assessment may contain a high level of detail in some cases, such as for an integrated electronic flight instrument system. However, many installations may only need an informal review of the system design by the applicant for the benefit of the FAA certification personnel to determine the criticality of the functions performed by the system. The purpose of the preliminary hazard analysis is to identify the critical and

essential functions and the systems which must operate properly to accomplish these functions. Once the criticality of a system has been established, additional techniques which might be useful in determining compliance with FAR 25.1309(b) are more easily identified.

b. Analysis of systems which perform non-essential functions.

(1) Although a preliminary hazard analysis has been accomplished, and it has been determined that a particular system performs only non-essential functions, this is not sufficient for demonstrating compliance with the requirements of FAR 25.1309(b). It is also necessary to determine if failures of the system could contribute to a failure condition involving any essential or critical function.

(2) In general, the installation of a non-essential system should be accomplished in a manner which insures its independence and isolation from other systems in the airplane which perform critical or essential functions. If a review of the design based on good engineering judgment determines that system faults cannot affect essential or critical functions, then no further analysis is necessary. If the installation does not have satisfactory isolation from systems which perform essential or critical functions, or if the system complexity is such that a design review alone cannot adequately establish that such isolation has been achieved, then the system should be analyzed using more rigorous methods, some of which are identified in paragraphs 6c and 6d, below.

(3) Special care must be taken with systems that perform non-essential functions which provide information for use by the flight crew, such as engine performance data systems. Systems of this type, which are not required by regulation and also are non-essential, may have hazardous failure modes which provide misleading information to the flight crew without warning. These systems may have to be analyzed as a system which performs an essential function.

(4) Typically, systems such as galleys, position lights, public address systems, and interior cabin lights, to name a few, should be certificated based on a design review alone without the need of a formal failure analysis. Note that some systems required by regulation may be found to perform non-essential functions using the criteria of this advisory circular. Equipment such as transponders, position and anticollision lights, altitude alerting systems and ground proximity warning systems, are required for various operations and airplanes by regulation for safe and expeditious use of the airspace, but loss of this type of equipment does not create a serious hazard to the airplane or prevent its continued safe flight and landing and may therefore be considered to perform non-essential functions.

c. Analysis for failure conditions involving systems which perform essential functions.

(1) Failure conditions which affect essential functions should be improbable. Satisfactory service history of the equipment under analysis or similar units will be acceptable for showing compliance. Compliance may also be shown by a quantitative reliability analysis using failure rates from an acceptable industry standard or actual equipment failure rate data. An

acceptable probability level within the defined improbable range should be agreed upon with the FAA for a particular failure condition. Determination of the acceptable probability level should be based on an inverse relationship between the probability of the failure condition and the severity of its effect on airplane safety. This is not meant to imply that a numerical analysis will always be required to show compliance with an agreed-to level.

(2) Many units which perform essential functions have dual or greater redundancy. If redundancy exists and there is some evidence to indicate the satisfactory reliability of the redundant subsystems, no further analysis is necessary. For complex systems, a failure modes and effects analysis may be necessary to verify that redundancy actually exists, and to show that the failure modes of the system do not have an adverse effect on other essential or critical functions. A complete quantitative safety analysis will not usually be necessary.

(3) If failure modes are found to exist which result in failure conditions, these failure conditions should be shown to be improbable or extremely improbable, depending upon the criticality of the affected function. However, failure conditions resulting from single faults will not usually be accepted as being extremely improbable. In unusual cases, a failure condition resulting from a single fault can be assessed as extremely improbable if it can be shown that based upon construction, installation and experience such a fault need not be considered as a practical possibility.

d. Analysis for failure conditions involving systems which perform critical functions.

(1) A quantitative safety analysis will generally be necessary for each failure condition identified by the preliminary hazard analysis that would prevent the continued safe flight and landing of the airplane. Such failure conditions should be extremely improbable. If a quantitative safety analysis is required, the analysis may include the following:

(i) FAULT TREE ANALYSIS

(ii) FAILURE MODES AND EFFECTS ANALYSIS--An inductive bottom up analysis which determines what happens to the system upon single failures of its individual parts. These failure modes are used as the bottom level events of the fault tree analysis.

(iii) PROBABILITY ANALYSIS--Determines the probability of the single faults used as bottom level events of the fault tree analysis from failure rate data and exposure times to both active and hidden failures. The probability of all event conditions in the fault tree analysis will then be calculated from this data. The fact that maintenance or flight crew checks will be performed throughout the life of the system is relevant to quantitative analysis. When exposure times applicable to probability calculations for critical functions are affected by flight crew checks or maintenance inspection intervals, these time intervals and check procedures should be clearly specified in appropriate documents. Required flight crew member actions should be

specified in the limitations section of the Airplane Flight Manual. Required maintenance procedures and inspection intervals should be included in the Airworthiness Limitations section of the Instructions for Continued Airworthiness. The required maintenance procedures and inspection intervals should also be made known to the FAA Maintenance Review Board (MRB) which develops the initial aircraft maintenance program. The specific data will be used in determining the initial maintenance requirements for inclusion in the MRB document. Changes to the Airworthiness Limitations section as service experience is gained on the airplane must be approved by the FAA Transport Airplane Certification Directorate. An owner or operator of the airplane may request that alternative inspection intervals and related procedures be set forth in an operations specification approved by the Administrator under Parts 121, 123, 125, 127 or 135 or in accordance with an inspection program approved under FAR 91.217(e). For very simple installations, it may be possible to successfully analyze a failure condition involving a critical function without using the detailed formal procedures outlined above. In general, the simultaneous failure of two reliable independent systems, each of which has dual redundancy, is expected to be extremely improbable.

(2) The increasing use of digital avionics systems in aircraft has focused attention on the probability of failure conditions caused by errors in the specification of system requirements or implementation of system design. Of particular concern are errors in the computer programs used by software based digital equipment. This advisory circular has outlined the use of quantitative safety analysis for evaluating some types of systems which perform critical and essential functions. At this time, valid quantitative methods for evaluating the probability of system errors have not been identified by the aviation industry or the Federal Aviation Administration. However, a design methodology for software based systems has been developed by the Radio Technical Commission for Aeronautics (RTCA). These recommendations are contained in RTCA Document DO-178, are accepted by the FAA, and should be followed for software based systems which perform essential and critical functions.

e. The analytical techniques outlined in this section provide acceptable techniques, but not the only technique for determining compliance with the requirements of FAR 25.1309(b). Other comparable techniques exist and may be proposed by an applicant for use in any certification program. However, these methods should be proposed to the FAA certificating office early in the program. Early agreement between the applicant and the Federal Aviation Administration should be reached on the methods of analysis to be used, identification of critical functions, and assumptions to be used in the acceptance of the proposed analysis.

f. The analysis should be clearly documented. All assumptions, sources of reliability data, failure rates, system functional type (critical, non-essential, essential), etc. should be concisely documented for ease of review. To the extent feasible, the analysis should be self-contained.

7. RECOMMENDATION. The purpose and intent of this advisory circular is to provide guidance. Terms and methods of analysis which may be utilized in

9/7/82

demonstrating compliance with FAR § 25.1309 are included. If additional explanation or discussion is desired, contact the Transport Airplane Certification Directorate, Aircraft Certification Division, Regulations and Policy Office, ANM-110, 17900 Pacific Highway South, C-68966, Seattle, Washington 98168, or phone 206-764-7051.



Charles R. Foster
Director, Transport Airplane Certification Directorate

9/7/82

AC 25-1309-1
Appendix 1

APPENDIX 1. BACKGROUND INFORMATION FOR CONDUCTING FAILURE ANALYSES

1. INFORMATION SOURCES. For those unfamiliar with the concepts of systems analysis in general and fault tree analysis in particular, the U.S. Nuclear Regulatory Commission has published NUREG-0492, titled "Fault Tree Handbook." This document provides a detailed description of this method of analysis which has been used successfully by various manufacturers to determine the probability of a particular failure condition for FAA certification programs. The handbook also provides a bibliography of additional books, articles, and papers on the subject of reliability. The format of quantitative analyses which use NUREG-0492 as a guide will be acceptable to the FAA. Copies of this document can be obtained from the National Technical Information Service, or from:

GPO Sales
Division of Technical Information and Document Control
U. S. Nuclear Regulatory Commission
Washington, D.C. 20555

If an equipment manufacturer does not have an acceptable record of service experience necessary to estimate the reliability of an item of electronic equipment, MIL-HDBK-217 (Reliability Prediction of Electronic Equipment) may provide a satisfactory means to perform this estimate.

A manufacturer or operator may record service history information on the basis of hours of flight time (block hours), flying hours, operating hours, cycles, etc. This information may be converted into hours of flight time by the application of appropriate conversion factors.

2. IDENTIFICATION AND EVALUATION OF CRITICAL FUNCTIONS. The preliminary hazard analysis is noted in this advisory circular as a means of identifying critical and essential functions and the systems which must operate properly to accomplish these functions. Critical functions are generally those for which no satisfactory substitute is available and which must be accomplished for continued safe flight and landing of the airplane.

Examples of some systems which perform critical functions that have been identified on various transport category airplanes are listed below. This list is only provided to illustrate the types of functions which may be critical. Each airplane model must be examined to determine what functions are critical.

- a. The primary flight control system.
- b. Hydraulic power for airplanes with powered flight control systems and no manual reversion.
- c. Secondary flight control systems, if failure of these systems can result in uncontrolled flight.
- d. Engine control system elements that affect all engines simultaneously.

9/7/82

e. For airplanes certificated for flight in instrument meteorological conditions, the total systems and displays which provide the flight crew with any of the following:

- (1) Attitude Information
- (2) Altitude Information
- (3) Airspeed Information

f. Automatic landing system for use in low visibility landings.

When determining the extent of the analysis to be conducted for failure conditions involving systems which perform critical functions, a number of factors should be considered. A failure modes and effects analysis should be performed on a system which performs critical functions if its complexity is such that the effects of its failure modes are not obvious. A quantitative analysis is normally needed only when systems which perform critical functions differ significantly in design or application, as listed below, from existing systems which satisfactorily perform these functions.

- a. Technology
- b. Interrelationship with other systems on the airplane.
- c. Relationships between the system and critical characteristics of the airplane.
- d. Complexity

For example, systems performing critical functions, such as a mechanical control cable system for primary flight controls with dual redundancy or a hydraulic power system with triple or quadruple redundancy used by a fully powered flight control system would not necessarily be the subject of a quantitative analysis. However, even if these systems were similar in design to those in current service, they would normally be the subject of a failure modes and effects analysis.

When it has been determined that a quantitative analysis should be conducted, the following should be considered:

a. Human errors in operation and maintenance. The design of systems which perform critical functions should be such that failure of the systems do not require flight crew action to prevent the failure condition beyond the tasks normally required to fly the airplane, or that system failures which require flight crew intervention provide a clear and unmistakable annunciation to alert the flight crew that the failure has occurred and the subsequent flight crew member actions necessary to prevent the failure condition can be satisfactorily accomplished. The failure annunciation and the required flight crew member actions should be evaluated by FAA flight test pilots to determine if the necessary actions can be satisfactorily accomplished in a timely manner without

9/7/82

AC 25.1309- 1
Appendix 1

exceptional pilot skill or strength. If satisfactory action by the flight crew is doubtful, then reliance on flight crew intervention should not be assumed in determining the probability of the failure condition. If the evaluation determines that satisfactory intervention can be expected from a properly trained flight crew, then the occurrence of the failure condition has been prevented.

In a similar manner, an assessment should be made of the design and of the maintenance instructions with the object of eliminating the possibility of maintenance errors which could result in a failure condition. Maintenance tasks which are required should be evaluated to determine if they can be reasonably accomplished. For the purposes of a quantitative analysis, the satisfactory accomplishment of identified maintenance tasks should be assumed to be one (1). The FAA believes that a numerical assessment of the probability of human error on the part of the flight crew or maintenance personnel is not appropriate for the purposes of conducting a design analysis.

b. System Independence and Redundancy. The most often encountered difficulty with quantitative analyses presented to the FAA has been the improper treatment of events which are not mutually independent. The probability of occurrence of two events which are mutually independent may be multiplied to obtain the probability that both events occur using the formula:

$$P(A \text{ and } B) = P(A)P(B).$$

This multiplication will produce an incorrect solution if A and B are not mutually independent. Often a quantitative analysis will be defective because a single failure will be included as a primary event at more than one location and then improperly combined with itself in computing the probability of the top event of a fault tree.

Another persistent problem is the identification of common failure modes which simultaneously affect the operation of two or more separate systems which otherwise are independent. The loss of electrical power, hydraulic power, or cooling may often result in common failure modes. A failure modes and effects analysis is often useful in identifying common failure modes.

APPENDIX E

**ADVISORY CIRCULAR AC: 20-115
RADIO TECHNICAL COMMISSION FOR AERONAUTICS
(RTCA) DOCUMENT DO-178**

APPENDIX E

ADVISORY CIRCULAR AC: 20-115
RADIO TECHNICAL COMMISSION FOR
AERONAUTICS (RTCA) DOCUMENT DO-178



U.S. Department
of Transportation
Federal Aviation
Administration

Advisory Circular

Subject: Radio Technical Commission for
Aeronautics (RTCA) Document DO-178

Date: 9/3/82
Initiated by: AWS-120

AC No: 20- 115
Change:

1. PURPOSE. This Advisory Circular (AC) calls attention to the Radio Technical Commission for Aeronautics (RTCA) Document DO-178, "Software Considerations in Airborne Systems and Equipment Certification." It discusses how the document may be applied with FAA Technical Standard Order (TSO), Type Certification (TC), and Supplemental Type Certification (STC) authorizations.
2. RELATED FAR SECTIONS. FAR Parts 21, 23, 25, 27, 29, and 33.
3. BACKGROUND. RTCA Document DO-178 was issued November 1981 to establish software considerations for developers, installers, and users when the aircraft equipment design is implemented using microcomputer techniques. It is expected that future avionics designs will make extensive use of this technology. The RTCA document outlines verification, validation, documentation, and maintenance procedures to be used in microcomputer systems.
4. USE OF DO-178 PROCEDURES. An applicant for a TSO, TC or STC authorization for any electronic equipment or system employing digital computer technology may use the considerations outlined in RTCA Document DO-178 as a means, but not the only means, to secure FAA approval of the digital computer software. The FAA may publish Advisory Circulars outlining "criticality categorization" for specific FAR's. Those AC's may differ from and will take precedence over application of DO-178, § 5.1.
5. WHERE TO OBTAIN COPIES OF RTCA DOCUMENT DO-178. Copies of DO-178 may be obtained from:

RTCA Secretariat
Suite 655
1717 H Street, NW.
Washington, D.C. 20006

M. C. Beard
Director of Airworthiness

END

FILMED

10-83

DTIC